National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

NL-339 (r. 88/04) c

Canada

UNIVERSITY OF ALBERTA

Identification Issues in Long Range Predictive Control

BY

David Stephen Shook

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

Doctor of Philosophy

Department of Chemical Engineering

EDMONTON, ALBERTA

Spring, 1991

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-66814-8

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: David Stephen Shook

TITLE OF THESIS: Identification Issues in Long Range Predictive Control

DEGREE: Doctor of Philosophy

YEAR THIS DEGREE GRANTED: 1991

Permission is hereby granted to UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(SIGNED) David Shook

306 Albert Ave.

Saskatoon, Sask.

S7N 1G1

DATED: Apr. 24 19 91

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled *Identification Issues in Long Range Predictive Control* submitted by David Stephen Shook in partial fulfilment of the degree of Doctor of Philosophy in Chemical Engineering.

.....
Prof. S. L. Shah (supervisor)

.....
Prof. R. K. Wood

.....
Prof. D. G. Fisher

.....
Prof. V. G. Gourishankar

.....
Prof. P. J. Gawthrop

Date APRIL 24 1991

This work is dedicated

to the memory of Margaret Dempsey,

to my friends,

and to my new family.

Abstract

The field of adaptive control has been growing and gaining momentum for almost twenty years. Many sophisticated adaptive control methods exist but the long-term quality of control always relies on the quality of process identification.

The role of process identification in adaptive implementations of Generalized Predictive Control (GPC) was examined in this work. An overall process control objective — including both process identification and control — was proposed. The intention was for both identification and control to be parts of an adaptive controller designed as a complete unit. The identification and control parts would then work toward the same goal of good control. When the overall control objective was chosen such that GPC was a logical control method, the corresponding control-relevant identification objective was found to be different from the usual least squares objective.

The control-relevant identification objective makes use of long range predictions, as does GPC. To illustrate this property. the new identification method was named Long Range Predictive Identification, or LRPI. The exact solution can be found only using iterative methods (e.g. Newton-Raphson), but an approximate (asymptotically equal) recursive solution method was found that uses an adaptive data pre-filter and normal recursive least squares.

The LRPI pre-filter depends on the process and noise models and the GPC controller tuning. Under the conditions studied, the filter had frequency domain properties similar to those of *ad hoc* pre-filters used in conjunction with RLS by other researchers. Adaptive GPC using LRPI provided control performance at least equal to that when RLS was used with an *ad hoc* filter, with the additional advantage that the LRPI filter is chosen automatically by the identification method, and therefore need not be specified by the user.

An additional concern arises when an adaptive controller is used for both

feedforward and feedback control. Care must be taken in such cases that the process identification task does not inappropriately discard information in uninformative feedforward variables. Non-directional forgetting factors will inadvertently discard such information under conditions where "directional" forgetting factors will not. Directional forgetting methods are therefore recommended over exponential methods.

Acknowledgements

This book is the product of many people, even though only one name appears as the author. Two people in particular made it happen: my wife, Kath, and Coorous Mohtadi. Kath's sense of humour and support were invaluable, and Coorous's technical advice and broad hints showed me the way over and over again. I would also like to thank Dr. Sirish Shah and my parents for their support (both moral and financial) and the large number of prods they applied to bring this project to an end.

In addition, the number of friends, both within and beyond the Department of Chemical Engineering, who contributed to this work is impossible to count. I have enjoyed my time with all of you. It would be unfair to shine the light on a few: the absence of a mention would darken by comparison. Thank you all.

Contents

1 Introduction 1

2 Commercial Adaptive Control Software 7

2.1 Functional Specifications 8

2.2 Controller Description 10

2.2.1 Process Communication 12

2.2.2 Parameter Estimation 14

2.2.3 Controller Calculation 19

2.2.4 Database Management 23

2.3 User Interface 24

2.4 Testing and Verification 28

3 Experimental Verification and Performance 30

3.1 Equipment Description 31

3.2 Experimental Performance of Software 34

3.2.1 Operation in the Presence of Communication Difficulty 43

3.2.2 Feedforward Control Performance 47

3.3 Weaknesses of the Existing Algorithm 47

4 Identification for Adaptive GPC 51

4.1 Parameter Estimation Methods 53

4.2 Overall Control Criterion 59

4.2.1 Dual Control 61
4.2.2 Certainty Equivalent Formulation 64
4.2.3 An Alternative Formulation 67

5 Long Range Predictive Identification **69**

5.1 The LRPI Objective 69
5.2 Exact Solution of the LRPI Problem 78
5.2.1 Batch Solution 79
5.2.2 Recursive Solution 82
5.3 Implementation Through Adaptive Filtering 84
5.3.1 Frequency Domain Analysis of Cost Function 84
5.3.2 Spectral Factorization 89
5.3.3 Properties of the Adaptive Filter 91
5.3.4 Stochastic Extensions 98
5.3.5 On-line LRPI Implementation 100

6 Evaluation of LRPI **103**

6.1 Simulation Examples 103
6.1.1 Open Loop 104
6.1.2 Closed Loop 105
6.2 Experimental Studies 111
6.3 Conclusions 117

7 Nonminimal Predictive Control **124**

7.1 Design Philosophy 124
7.2 The NPC Predictor 127
7.3 The NPC Noise Model 129
7.3.1 Control Calculation 131
7.4 Parameter Estimation for NPC 132

7.5 Simulations 134
7.6 Conclusions 137

8 Forgetting Methods for Adaptive GPC **138**
8.1 Forgetting in Recursive Least Squares 139
8.2 Parameter Confidence Bounds 142
8.3 The Effects of Forgetting on Parameter Confidence 148
8.3.1 Implications to Feedforward Adaptive Control 158
8.4 Experimental Studies 163
8.5 Conclusions 171

9 Conclusions and Recommendations **175**
9.1 Conclusions 175
9.2 Recommendations for Future Work 177

References **181**

A Lrpiest User's Guide **187**
A.1 User's Overview 187
A.1.1 Scope of This Manual 187
A.1.2 Algorithms and References 188
A.1.3 Use of the program 188
A.2 Programmer's Overview 191
A.2.1 Common Routines 192
A.2.2 Forgetting Factor Implementation 193
A.2.3 Filtering Routines 195

B MATLAB Files **198**
B.1 Introduction 198
B.2 LRPI .M Files 198

B.2.1 Batch Calculation 198
B.2.2 Recursive Calculations 200
B.3 Forgetting .M Files 202

List of Figures

2.1 Schematic of the GPC controller architecture 11

2.2 Model Structure Page 26

3.1 Schematic Diagram of Experimental Heater System 32

3.2 Experimental Open Loop Response to Control Action Change 34

3.3 Experimental Open Loop Response to Disturbance Change 35

3.4 Closed Loop Performance of Adaptive GPC for Run 1 37

3.5 Prediction Errors for Run 1 38

3.6 Controller Transfer Function Parameters for Run 1 40

3.7 Closed Loop Performance of Adaptive GPC for Run 2 41

3.8 Controller Transfer Function Parameters for Run 2 42

3.9 Closed Loop Performance of Adaptive GPC for Run 3 43

3.10 Controller Transfer Function Parameters for Run 3 44

3.11 Process Output and Communication Errors 46

3.12 Closed Loop Performance for Run 4 48

3.13 Prediction Errors for Run 4 49

5.1 Graphical Interpretation of Predictions in LRPI Cost 71

5.2 Contour Plot of LRPI Cost as a Function of Parameter Estimates . . 74

5.3 Contour Plot of LS Cost as a Function of Parameter Estimates 76

5.4 Step Responses of LS and LRPI Models 77

5.5 Variation of the Frequency Response of L with Process Model Pole Position 94
5.6 Variation of the Frequency Response of L with N_2. 96
5.7 Comparison of L and *ad hoc* Filters 97
5.8 Variation of the Zero of T_{eq} with N_2. 99
5.9 On-Line Implementation of Adaptive GPC with LRPI 101

6.1 Step Response of Rohrs *et al.* Plant and First Order Models for Sampling Frequency of 3 Hz 104
6.2 Step Response of Rohrs *et al.* Plant and First Order Models for Sampling Frequency of 20 Hz 106
6.3 Closed Loop Poles for GPC and LRPI with Different Values of N_2 . . 107
6.4 Response of Self-tuning GPC for First Order Model, $T = 1 - 0.9q^{-1}$ and Standard RLS 108
6.5 Response of Self-tuning GPC for First Order Model, $T = (1-0.8q^{-1})^2$ and Standard RLS 109
6.6 Response of Self-tuning GPC for First Order Model, $T = 1 - 0.8q^{-1}$ and LRPI 110
6.7 Experimental Equipment 112
6.8 Process Input and Output for GPC with LS Estimator. 114
6.9 Process Input and Output for GPC with LS Estimator and an *ad hoc* Filter 115
6.10 Process Input and Output for GPC and LRPI 116
6.11 Parameter Trajectories for GPC with LS Estimator. 118
6.12 Parameter Trajectories for GPC with LS Estimator and an *ad hoc* Filter 119
6.13 Parameter Trajectories for GPC and LRPI 120
6.14 Evolution of LRPI Filter 121
6.15 Comparison of GPC Using Low and High Order Models 122

7.1 Comparison of GPC and NPC: Noise Free Case 135
7.2 Comparison of GPC and NPC: Measurement Noise Present 136

8.1 95% Confidence Limits for a Single Parameter Estimate 144
8.2 95% Confidence Limits for Two Parameter Estimates 145
8.3 Evolution of Parameters and 95% Confidence Region for RLS 147
8.4 Evolution of Parameters and 95% Confidence Region for Exponential Forgetting 150
8.5 Evolution of Parameters and 95% Confidence Region for Directional Forgetting, $\alpha = 0.6$ 154
8.6 Evolution of Parameters and 95% Confidence Region for Directional Forgetting, $\alpha = -0.2$ 155
8.7 Initial Three Dimensional 95% Confidence Region 160
8.8 95% Confidence Region for Ydstie's Forgetting Factor 161
8.9 95% Confidence Region for Constant Trace Forgetting Factor 162
8.10 95% Confidence Region for Kulhavý's Directional Forgetting Factor . 163
8.11 95% Confidence Region for Hägglund's Directional Forgetting Factor 164
8.12 Process Inputs and Outputs for Exponential Forgetting with GPC . . 167
8.13 Process Inputs and Outputs for Directional Forgetting with GPC . . 168
8.14 Exponential Forgetting Factor Value for Exponential Forgetting with GPC 169
8.15 Forgetting Factor Values for Directional Forgetting with GPC 170
8.16 Confidence Bounds for Directional Forgetting and GPC 172
8.17 Confidence Bounds for Exponential Forgetting and GPC 173

List of Tables

5.1 Summary of Cost Function Information for Example 5.2 75

A.1 System Data Table Mnemonics Required for Lrpiest 189

A.2 Values for EstON 190

A.3 Source Files for Lrpiest 192

Chapter 1

Introduction

The field of self-tuning control has been growing for nearly twenty years. Ever since the ground-breaking work of Åström and Wittenmark (1973), many different self-tuning controllers have been proposed by numerous researchers. This work is concerned with the implementation of a particular self-tuning controller: the generalized predictive controller, or GPC of Clarke *et al.*(1987a,b). GPC has been successfully used in a number of experimental applications, and is distinguished by its ease of configuration and flexibility. The implementation of a self-tuning controller nevertheless requires the specification of a surprisingly large array of parameters, especially for a controller as general as GPC. It is one goal of this work to rationalize and simplify these choices.

A self-tuning controller may be divided conceptually into two parts: a process identification scheme and a model-dependent controller. It is the role of the process identification scheme to provide the controller with an up-to-date mathematical description of the process, and the controller must then produce the appropriate control action.

The identification scheme identifies the dynamic characteristics of the process to be controlled, through analysis of process inputs and outputs. It also updates the description (or model) in the face of changes in the true process. The control

action is then chosen to make the model output behave in a user-specified manner. The interaction between process identification and the controller is fairly involved and the analysis of adaptive systems is necessarily complex. Nevertheless, the process identification scheme is usually the weak link in an adaptive controller. This is because the controller is typically such that with an accurate or "good" model the closed loop response will be well-behaved. With a sufficiently inaccurate (or "poor") model control performance will be unsatisfactory. The process model is a product of the identification scheme, and different identification schemes will produce different models. Furthermore, the identification schemes themselves have tuning parameters that must be chosen by the user. Moreover, adaptive controllers are typically used by engineers, who are more at ease with the mathematics used for the control calculations than with the statistics required to understand the subtleties of process identification.

The work described in this thesis resulted from a need for more robust, more easily tuned adaptive control algorithms. Ideally, an adaptive controller should start up from an initial condition with no knowledge about the process and then operate safely and in some sense optimally without operator intervention. Failing that, and more realistically, an adaptive controller would be useful if it could achieve and maintain high-quality control (after initial operator tuning) in spite of changes in the process dynamics. In other words, the controller may not be self-commissioning, but it should be easy to commission and once commissioned it should be self-maintaining and not require periodical retuning.

Scope and Objectives

This thesis explores the implementation of a single adaptive controller: the generalized predictive controller of Clarke *et al.* (1987a,b), or GPC. GPC calculates the control action using a process model and is therefore only one half of a usable adaptive controller. The process identification scheme typically used with GPC and

examined in this work is Recursive Least Squares, or RLS. The objective of this work is to make adaptive GPC simpler to implement and easier to use.

GPC provides a mechanism for minimizing the predicted control errors over a finite time horizon in the future. It has been shown to work well in the presence of measurement noise and model-plant mismatch (McIntosh, 1988), conditions so common in practice as to be almost universal. The chief advantage of GPC over other robust self-tuning controllers is its ease of initial configuration. Given an adequate model, there is a default configuration which typically produces a stable, detuned closed loop. The user can then "tighten" control incrementally using one or two controller parameters (see e.g. McIntosh, 1988). For high-speed control (e.g. control of robot arms, M. Lambert, 1987) where a high-order model is used, configuration is more complicated than for process control applications but still tractable.

The method of Recursive Least Squares (RLS) is discussed extensively throughout the self-tuning control literature (e.g. Goodwin and Sin, 1984, Ljung and Söderström, 1983 and Ljung, 1987). RLS is simple to implement and has many attractive theoretical properties, but the original algorithm has been modified many times and in many different ways. Most of the modifications have been made in response to one of two problems: either the Least Squares estimated model does not provide good control or the estimated model does not track parameter changes. The first problem has been addressed using data filtering and noise modelling and the second problem has spawned the whole field of data forgetting.

The filtering applied to the measurements has been shown to have a profound effect on the identified model. Many of the papers in Shah and Dumont (1990) concur on the fact that filtering is required. There is no unanimity on *what* filtering is required, although the consensus is that some form of band-pass filter is usually appropriate. Curiously enough, the filtering required for identification is often different from the filtering required by the controller. This counterintuitive result has been discussed by a number of researchers, including McIntosh (1988), Mohtadi (1990),

and Wittenmark (1990) but the explanation has been heuristic and *ad hoc*.

The major contribution of this work is the presentation of the theoretical reasons for different data filtering for identification and control. The use of different filters is justified by the identification objective: to provide high-quality *control*. A new **control-relevant** identification method is developed that is shown to be very similar to RLS and is in fact realized by use of an adaptive data pre-filter in cascade with the RLS estimation scheme.

RLS is derived from a batch procedure designed to find the single "best" model in the least square sense. Batch least squares has no facility for changing the model but Recursive Least Squares is often used in conditions where the "best" model changes over time. Data forgetting was introduced to accommodate such changes. Forgetting ensures that the estimation scheme gives higher "weighting" to the most recent data since these data provide the most up to date cause and effect correlations. The choice of a forgetting method can be complicated and is made more so when there are several inputs to the process, as when feedforward control is used with feedback. The issue of "directional" versus "exponential" forgetting was addressed during this study. Directional forgetting was found to possess qualities that are necessary when an adaptive controller is used in a combined feedforward and feedback role. Directional forgetting is particularly important when the estimator must retain information in the absence of current input excitation in some input signals.

In brief, the objective of this work is to facilitate the implementation of adaptive control, specifically GPC. The contributions of this work are:

1. the implementation of an adaptive controller in an industrial environment,
2. the development of a control-relevant identification method for GPC, suitable for extension to other model-based predictive controllers,
3. the identification of directional forgetting as the logical choice for adaptive

feedforward and feedback control.

Organization

To evaluate the implementation of adaptive GPC, adaptive controller software was written and used to control a pilot scale process using standard industrial control equipment at the Fort Saskatchewan site of Dow Chemical Canada Inc. A state of the art adaptive controller, discussed in Chapter 2, was implemented in early 1988. During tests at the Dow site, some shortcomings of the implementation were noticed in spite of generally good performance. The program verification and conclusions drawn from the tests are described in Chapter 3.

Two issues in particular were raised. Specifically, the noise model in GPC has been used to cover a multitude of sins, and has thus been exceedingly difficult to choose. In addition, the problem of model updating is complicated when both feedforward and feedback signals are used to calculate the control action, as then there are several inputs to the process which may contain different amounts of information.

The noise model in GPC has heretofore been used to describe the effects of unmeasured process inputs, measurement noise and unmodelled dynamics. There has been a certain amount of confusion in the adaptive control literature regarding whether or not the noise model should include the effects of model-plant mismatch. This has led to incoherent treatment of the noise model. The concept of an overall adaptive control objective is used in Chapter 4 to reduce this confusion. Rather than choosing a process identification scheme independently of the control algorithm, the overall control objective prescribes the use of a "control-relevant" identification strategy.

The overall control goal is shown to produce a new process identification objective that is consistent with the long-range predictive nature of GPC. Rather than using the standard least squares approach, a long range predictive identification method is needed to provide the long range predictions needed by GPC. This

method, called LRPI, is discussed in great detail in Chapter 5. Its implementation, especially in recursive (on-line) applications, is computationally intensive, and frequency-domain arguments are used to cut the Gordian knot. The final on-line implementation is remarkably simple. Both simulation and experimental studies were conducted, and results are presented with analysis in Chapter 6.

An alternative approach to the overall adaptive control problem is the Nonminimal Predictive Controller (NPC) of Lu and Fisher (1990). It is discussed in Chapter 7. It is an interesting method which uses a different predictor structure to circumvent the extra filtering required for LRPI. NPC is of considerable theoretical interest, but there are severe drawbacks to implementation, as is pointed out in Chapter 7.

Chapter 8 addresses the question of persistent excitation in the multivariable context of feedback plus feedforward (FB+FF) or multi-input, single output (MISO) adaptive control. A MISO adaptive controller measures a number of feedforward variables as well as the process output and attempts to find the statistical relationships among the different process measurements. In adaptive control, data must be discarded periodically if the controller is to track changes in the process (Ljung and Söderström, 1983). The problem of data forgetting in MISO adaptive control is discussed, and a directional forgetting approach, such as that of Kulhavý (1987) or Hägglund (1983) is recommended.

Conclusions and recommendations comprise Chapter 9.

Appendix A contains documentation for the parameter estimation program used for the experimental studies described in Chapters 6 and 8. The MATLAB script files used for simulations and analyses are included in Appendix B.

Chapter 2

Commercial Adaptive Control Software

This chapter describes a software package developed for Dow Chemical Canada Inc. The software was designed to make adaptive control technology available to Dow personnel for evaluation at the Fort Saskatchewan, Alberta site using their own control computers and processes. Initial work started on the project in January 1988, and the software was essentially complete by the end of October 1988. Experiments were conducted at Dow during May 1989, during which time some program maintenance was carried out and the software was demonstrated to Dow personnel at the site as well as those attending the 1989 Dow Global Advanced Control Meeting.

The software was written in FORTRAN for a VAX/VMS system and consisted of two main programs. The adaptive controller itself required over 2000 lines of source code, and the user interface required roughly 3500 lines of FORTRAN.

Development of the adaptive controller program took roughly three months; the process communication interface took another three months, and the user interface required another four months. The documentation (Shook and Shah, 1989) was completed in the following three months, including a number of revisions imposed by Dow for confidentiality. The manual, *The Adaptive GPC User's Guide*, describes

not only the installation and use of the program, but also contains an introduction to adaptive control theory. In this way it permits an engineer without an advanced control degree to use the software effectively.

The software was used to control a bench-scale process during final testing and demonstration. Some of the results are shown in Chapter 3.

The organization of this chapter is as follows. The functional specification of the software is given in Section 2.1. The adaptive controller itself is described in Section 2.2, and the user interface program is outlined in the following section, 2.3. A brief description of the testing and software verification procedure is given in Section 2.4.

2.1 Functional Specifications

The overriding functional requirement for the software was that it provide effective control in an industrial environment. The adaptive controller algorithm, chosen following discussion with Dow personnel, was GPC. The identification scheme chosen was Recursive Least Squares with a variable forgetting factor (Ydstie *et al.*, 1985). Additionally, there were the following functional requirements:

- The controller was to control a single loop.
- Feedforward control was to be used with feedback where appropriate; up to three feedforward measurements were to be used.
- All algorithms used were to be employed in their fully general form as published.

  All model polynomial orders, delays and initial parameter values were to be chosen by the user.

  For GPC all tuning parameters were to be available for the user to change on-line.

N_1, the initial prediction horizon,

N_2, the final prediction horizon,

NU, the control action horizon,

λ, the control action weighting,

$P(q^{-1})$, the output shaping transfer function,

$Q(q^{-1})$, the control action weighting transfer function,

$R(q^{-1})$, the setpoint response shaping transfer function,

$T(q^{-1})$, the noise model/disturbance rejection shaping filter.

For RLS, the model delays and orders could not be changed on-line, but all other user choices were to be available for on-line tuning:

σ, the estimated standard deviation of the noise,

N_0, the "asymptotic memory length" of the forgetting factor algorithm,

ϵ_{min}, the prediction error deadband,

ϵ_{max} the maximum permissible prediction error.

- The main adaptive control program was to produce a controller transfer function rather than the control action itself. This was to facilitate use of the program in a supervisory capacity or when the control action was implemented in a separate computer (as in a distributed control system).

- The main adaptive control program was to run in the background, at a user-specified sample rate. The control action would be calculated from the resulting transfer function in real time.

- The user interface was to be user-friendly and screen-oriented and permit real-time access to tuning parameters, process variables and diagnostic information.

The amount of information presented to the user on a given screen was to be limited to a few logically connected variables with descriptive text.

Motion among the specification screens was to be permitted in sequential order for initial specification, and in a random order for tuning of an existing controller.

User input was to be checked so polynomials would fit into the memory allocated at compile time for the arrays and to ensure a feasible controller was specified. ANSI FORTRAN has no facility for dynamic memory allocation, so all arrays had predefined, fixed limits.

On-line help was to be available, detailing the different commands.

- Communication between the user interface and adaptive controller was to be via a shared database file.
- All calculations were to be in single-precision FORTRAN, and the actual adaptation and controller calculation routines were to be in ANSI FORTRAN 77 for portability to other operating systems.
- All exceptions (run-time errors) were to be trapped and handled in an intelligent way to prevent the controller from terminating during a run.
- Comprehensive documentation was to be provided, including an introduction to adaptive control theory as well as a reference guide to installation and operation of the software.

2.2 Controller Description

The GPC controller consists of three main programs: the user interface, the real-time GPC controller design and the program which actually calculates the control action. The first two run in the host computer (VAX/VMS) and the last program – the

actual controller itself – runs in whatever computer is doing DDC and monitoring the process in real time, as shown in Figure 2.1. A number of supporting programs are provided for installation and analysis but strictly speaking they are not part of the GPC controller.

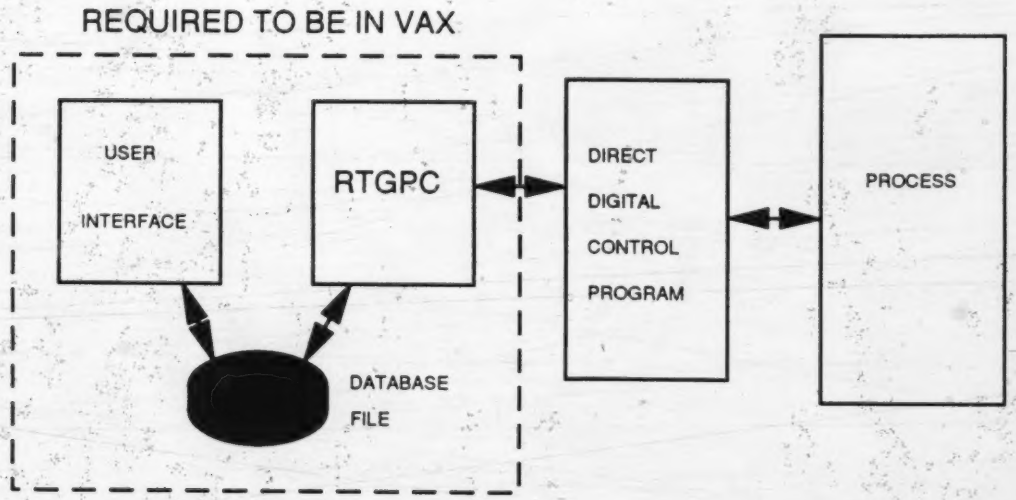

Figure 2.1: Schematic of the GPC controller architecture

The user interface allows the operator or engineer to monitor the controller and adjust its performance. It is described fully in Section 2.3. The user interface program, GPC.EXE, communicates with the real-time controller coefficient calculation program, RTGPC.EXE (also called the real-time executive) through a shared database file, and RTGPC communicates with the DDC program in an implementation-specific manner which is Dow Confidential and therefore beyond the scope of this thesis.

The entire software package may be installed on any VAX computer operating under VMS version 4.7 or later. A run-time license for FMS (the VMS Forms Management System) is required for the user interface. The control action calculation is

performed in Dow's proprietary Process Control System (d'PCS). Because of the use of system-dependent timing and screen manipulation, the host computer must be a VAX operating under VMS. However, the control action may be calculated in the VAX (for example when controlling a simulation) or in a d'PCS, or in any other DDC computer that can be connected to a VAX. The need for a VAX does not restrict the usefulness of the software, since many industrial process control systems are directly connected to VAX supervisory computers. Construction of the interface to the DDC system was the responsibility of Dow personnel for reasons of confidentiality.

The real-time controller program, RTGPC, takes the user's specifications, and, through observing the process, calculates the GPC controller transfer function to best control the process. It is composed of four main sections: process communications, parameter estimation, controller calculation and database management. Each of these sections will now be discussed in detail.

2.2.1 Process Communication

Whenever a digital computer is used to measure and control a continuous process, the question of data filtering or smoothing arises. Digital signal processing is quite advanced, and there are now, for example, standard solutions to the problem of aliasing (Stanley *et al.*,1984). The actual data collection and input filtering (if any) is all done by hardware and software provided by Dow and all the details are proprietary. There is however one detail in the process communication that is not Dow Confidential and yet is worthy of discussion. The problem of imperfect communication or sensor failure must be addressed by the GPC software.

Because the control action is calculated in a different program, and possibly a different computer, communication delays and failures are possible. An academic control package may be permitted to hang or fail if it misses a measurement, but an industrial control package must continue without interruption. In fact, commu-

nication difficulties are most likely when there is an upset in the plant, and reliable control is most necessary at such times.

Missing data complicate not only control but identification, especially when the data are filtered, and the problem is most severe when the filter in question is autoregressive in nature. The filtering used by RTGPC is imposed by GPC and is autoregressive. A missing measurement theoretically affects the filtered value for *infinite* time. In effect, the filtering would have to start again each time a single measurement is corrupted if no steps are taken to fill the gap. Some method of filling the gap is required.

When RTGPC receives the process information from the DDC computer, the validity of each measurement is checked. Typically, the measurements consist of controlled variable (PV), analog output value (AO), and up to three feedforward signals (F1, F2, F3). The setpoint is also measured, but a missing setpoint has no effect. If even one of the other measurements is invalid, then information is missing, so there is not enough information to update the process model. The identification and controller updates are abandoned. The control action will therefore be calculated using the previous process model. The missing or invalid datum is replaced with an estimate of the missing value, and the estimate is used in the filtering calculations.

For all measurements except the process output (setpoint, feedforward variables and control action) the replacement used is the previous value. The control action is not calculated locally, and so is an input to RTGPC. The control action is replaced with its previous value because there is no way for RTGPC to be sure that the GPC-specified control action is actually implemented. Output limits and the opportunity for operator intervention (especially during process upsets) dictated this conservative approach.

The missing process output variable (PV) value is replaced, not with its previous value, but with the prediction given by the process model. This is the best prediction available (or else it would not be used for the purposes of control). More-

over, the prediction error, were this estimate used to update the model — which it is not — would be zero.

The use of these procedures resulted in the GPC program performing well even in the presence of frequent communication upsets, as can be seen in section 3.2.1.

2.2.2 Parameter Estimation

The GPC controller explicitly requires an ARIMAX model (Clarke *et al.*, 1987a,b). The model used in RTGPC is an extension of the ARIMAX model. Other exogenous inputs are permitted as feedforward variables:

$$A(q^{-1})y(t) = B(q^{-1})u(t-d-1) + D_1(q^{-1})v_1(t-d_{v_1}-1) + T(q^{-1})\frac{\xi(t)}{\Delta} \qquad (2.1)$$

where $y(t)$ is the process output (controlled variable) at time t,

$u(t)$ is the process input (manipulated variable) at time t,

$v_1(t)$is a measured process disturbance variable at time t,

ξ is a random, zero-mean, stationary Gaussian disturbance,

q^{-1} is the backshift operator, $q^{-1}u(t) = u(t-1)$,

Δ is the differencing operator, $1 - q^{-1}$,

d is the process delay (between control action and output) expressed in samples,

d_{v_1} is the delay of the process with respect to v_1.

$A(q^{-1})$, $B(q^{-1})$, $D_1(q^{-1})$, and $T(q^{-1})$ are polynomials in q^{-1} of arbitrary order. $T(q^{-1})$ is a fixed filter which may be considered as describing the character of the process noise (Clarke and Gawthrop, 1979).

The method of Recursive Least Squares is used to estimate the coefficients of $A(q^{-1})$, $B(q^{-1})$ and up to three $D(q^{-1})$ polynomials corresponding to three measurable disturbances. Recursive Least Squares has been discussed in great detail

elsewhere, e.g. Ljung and Söderström (1983), Ljung (1987), Åström and Wittenmark (1989), so most of the discussion here will be devoted to the special features of the algorithm used in RTGPC.

RTGPC limits all the polynomials in equation 2.1 to no more than ten elements each, and the total number of identified parameters cannot exceed twenty. This is more than sufficient for any likely application, as models of order higher than 3 are very rare in chemical engineering applications. The high order B and D polynomials do, however, come in useful when the process has a varying time delay.

Recursive Least Squares (RLS) was chosen as the parameter estimation method for a number of reasons. First, it has been widely researched and discussed, and there are many successful applications in the literature. No original theoretical work was required to produce an active yet stable identification scheme. Secondly, there are few tuning parameters to be chosen by the user. In addition programming is straightforward, except for the covariance matrix update, which is available in pseudocode form in one of the standard references on the topic (Ljung and Söderström, 1983).

RTGPC uses some modifications of the standard RLS method: a variable forgetting factor (Ydstie *et al.*, 1985), UDU factorization of the covariance matrix for numerical stability, and a minimum and maximum prediction error deadband to turn identification on and off. The deadband provides robustness to the parameter estimation method during long quiet periods of operation and times of large disturbances.

Data Forgetting

When the dynamics of the observed plant change over time, then the adaptive controller must change its parameters to keep up with the process. This can only happen if old data are discarded once they are no longer valid. One common way to discard

outdated data is to use a "forgetting factor." The variable forgetting factor algorithm of Ydstie *et al.* (1985) is implemented in RTGPC. This particular scheme is used because it was felt that it provided smoother parameter estimate trajectories than the alternative methods such as the constant trace approach of Sripada and Fisher (1987).

The forgetting factor, λ, is calculated each time step. It is calculated to retain a constant amount of information regarding the process. The amount of information retained at any time is measured by the "nominal memory length," N_t, which is defined by Ydstie *et al.* (1985) as a weighted sum of squares of *a posteriori* errors

$$N_k = \sum_{k=1}^{t} \sigma_{k/t}\, r_i^{-1} (y_k - \hat{y}_k)^2 \tag{2.2}$$

where $\sigma_{t/t} = 1$

$\sigma_{k/t} = \lambda_t \sigma_{k/t-1}$

r_i is the variance of the measurement noise at time i

$\hat{y}_k$ is the *a posteriori* prediction of y_k

The value of λ_t is chosen to maintain N_t constant and equal to the initial value given by the user, N_0. All the user need specify are the desired memory length, N_0, and the noise variance (in RTGPC, the standard deviation). The memory length can be interpreted as the number of observations which would contain the same amount of information as is retained in the covariance matrix. A long memory improves the certainty in the parameters, but adaptation is limited in speed. A short memory allows the parameters to change more quickly, but at the cost of some extraneous fluctuations due to noise. Fortunately the choice is not critical, and really just the order of magnitude of N_0 is important.

Prediction Error Deadband

When the prediction error is very small, as may be the case during long periods of quiet operation, the covariance matrix tends to grow slowly, since information is forgotten (albeit slowly) and little or no meaningful information is added. Under these conditions, it is possible for the covariance matrix to grow without bound. The rate of adaptation is tied to the magnitudes of the elements of the covariance matrix. If the elements are large, it is possible for adaptation to occur inappropriately whenever there is even a small upset. The solution is to disable estimation whenever the prediction is within a user-specified threshold of the measured value. Such a prediction error deadband greatly improves the robustness of the adaptation, but its choice can be tricky, and it has been shown to result in biased parameters. In practice the deadband is typically set to between one-half and one standard deviation of the observed noise. This way updates will take place only when there is a significant error.

Numerical Considerations

The numerical stability of the adaptation mechanism provoked a considerable amount of work. RLS is numerically sensitive to roundoff error, especially when single precision arithmetic is being used. Single precision provides $7\frac{1}{3}$ significant figures, which seems sufficient, but can cause problems when data is gathered over a very long time. Roundoff errors accumulate in the covariance matrix and parameter estimates. Bierman (1977) discusses the numerical properties of the covariance matrix update in great detail. For "small" identification problems (low order models, finite experiment), RLS often works correctly, but in the application under discussion, the use of single-precision arithmetic and the requirement for long-term stability called for a better algorithm.

The UDU factorization technique of Bierman is therefore used. The covariance matrix is stored and updated in a factored form. Since it is symmetric and positive-definite, the covariance matrix can be factored as $P(t) = U(t)D(t)U^T(t)$, where $P(t)$ is the covariance matrix at time t, $D(t)$ is a diagonal matrix of *positive* numbers, and $U(t)$ is an upper-triangular square matrix with all entries on the main diagonal occupied by 1.

The covariance update program is taken from Ljung and Söderström (1983), and then modified slightly to permit the use of scaling. The scaling method of Sripada and Fisher (1987) is included in an attempt to improve the conditioning of the covariance matrix update, but after considerable discussion it was only implemented as an option. Its use is *not* recommended.

Noise Effects

The form of the predictive model used by a controller has profound effects on the controller's disturbance rejection properties. The noise model represents an internal model of the process disturbances (Francis and Wonham, 1976). If the internal model is incorrect, then the controller may not be able to completely remove the effects of disturbances. For example, the original self-tuning controller of Åström and Wittenmark (1973) could not completely remove the effects of step-type disturbances. The noise model of the CARIMA structure (equation 2.1)

$$A(q^{-1})y(t) = B(q^{-1})u(t-d-1) + D_1(q^{-1})v_1(t-d_{v_1}-1) + T(q^{-1})\frac{\xi(t)}{\Delta}$$

contains integrated noise. The signal ξ is defined as zero mean and stationary, but the integrated signal ξ/Δ is non-stationary. This form of model results in a controller that can annihilate the effects of nonstationary inputs as long as the inputs are well-described as integrated stationary processes (step-type or Brownian motion disturbances).

The effect of the noise model on implementation is that the process measurements (inputs and outputs) must be filtered by the reciprocal of the noise model before being used to identify the other parameters of the model. GPC assumes that the noise model is fixed, but it could be made time-varying through the use of Extended Least Squares (ELS) (Panuska, 1968). The disadvantage of ELS is that the noise input can only be approximated and is replaced with a proxy. The noise-model therefore converges *extremely* slowly if at all.

The noise model only allows for the effects of some disturbances. Large deterministic disturbances or sensor failures (discrete events) cannot be handled by the CARIMA model if ξ is a white noise process. RTGPC accommodates these by using a maximum prediction error. If the prediction error exceeds the maximum, then it is assumed to be at least partly caused by a large deterministic disturbance, and the data are not used for identification, although they are for filtering. The data are treated as invalid in the same sense as in section 2.2.1.

2.2.3 Controller Calculation

The GPC control law implemented in RTGPC consists of the full algorithm described in the original papers (Clarke *et al.*, 1987a,b). The GPC cost function is shown below.

$$J_{GPC} = \sum_{j=N_1}^{N_2} \left(R(q^{-1})y_{sp}(t) - P(q^{-1})\hat{y}(t+j \mid t)\right)^2 + \lambda \sum_{j=1}^{NU} (Q(q^{-1})\Delta u(t+j-1))^2 \tag{2.3}$$

where

N_1 is the initial prediction horizon,

N_2 is the final prediction horizon,

NU is the control action horizon,

λ is the control action weighting,

$P(q^{-1})$ is the plant output weighting transfer function,

$Q(q^{-1})$ is the plant input weighting transfer function,

$R(q^{-1})$ is the setpoint weighting transfer function,

$y_{sp}(t)$ is the setpoint at time t,

$\hat{y}(t+j \mid t)$ is the prediction, at time t, of $y(t+j)$,

$u(t)$ is the control action input to the process at time t.

The prediction horizons N_1 and N_2 are limited to 20, and N_2 must be at least as great as $N_1 + NU - 1$. NU is limited to less than 10. The numerators and denominators of the three transfer functions, $P(q^{-1})$, $Q(q^{-1})$ and $R(q^{-1})$ are limited to fourth order. These limits are all well beyond any values a user is likely to choose. P, Q and R in particular are rarely used. There is enough flexibility in N_1, N_2, NU and λ to achieve a good quality of control without their use. They are, however, useful in some research applications, where GPC is configured as a different form of adaptive controller (e.g. the Self-Tuning Controller of Clarke and Gawthrop, 1979 uses the P filter, $N_1 = N_2 = d$, $NU = 1$).

The role of RTGPC is just to provide the controller transfer function, not to calculate the control action. For a given model and set of tuning parameters, GPC reduces to a transfer function. McIntosh *et al.* (1990), for example, have shown how it is calculated. In adaptive implementations this form has both drawbacks and benefits. The amount of programming effort required is significantly increased, and more calculations are required to form the transfer function than would be normally required to calculate the control action directly. Moreover, any time any part of the model or controller configuration is changed, almost the entire calculation needs to be re-done. If the model is not changed, then the number of calculations required is trivial, so for a slowly-changing process the use of a transfer function form may actually reduce the computational load.

As for the identification, the question of numerical robustness was raised for

the control action calculation. In the GPC calculation a $NU \times NU$ matrix must be inverted. Mohtadi (1986) discusses a number of issues pertaining to the inversion. If the process model is sufficiently ill-conditioned, there may be a serious loss of precision. For the case when $NU = 1$, the matrix is a scalar (in fact a sum of strictly positive values), so the calculation is trivial and the problem does not exist. For larger values of NU or for MIMO GPC the numerical stability of the matrix inversion method used can affect the control action.

In accordance with the recommendations in Mohtadi (1986) the direct UDU factorization method is implemented to perform the matrix inversion. It provides a reasonably robust answer with a minimum of extra programming. Since NU is expected to be 1 in almost all cases, with only occasional use of higher values, it is not considered necessary to resort to singular value decomposition or any of the other more difficult algorithms.

The direct UDU factorization approach takes into account the structure of the matrix to be inverted. Specifically, it is of the form $\mathbf{G}^T\mathbf{G} + \lambda\mathbf{Q_n}$ where $\mathbf{G}$ is a matrix of step response coefficients, g_i, arranged (Clarke *et al.*, 1987a) as:

$$\mathbf{G} = \begin{bmatrix} g_0 & 0 & 0 & \cdots & 0 \\ g_1 & g_0 & 0 & \cdots & 0 \\ g_2 & g_1 & g_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{N_2} & g_{N_2-1} & g_{N_2-2} & \cdots & g_{N_2-NU+1} \end{bmatrix} \tag{2.4}$$

and $\mathbf{Q_n}$ is another matrix, which depends on NU and the $Q(q^{-1})$ weighting transfer function. For the vast majority of cases, $Q(q^{-1})$ is equal to 1, and $\mathbf{Q_n}$ is equal to $\mathbf{I}$, the identity matrix. (In the case that N_1 is greater than 1, the first $N_1 - 1$ rows of $\mathbf{G}$ are removed.)

If the i^{th} row of $\mathbf{G}$ is called $\mathbf{r}_i$, then $\mathbf{G}$ and $\mathbf{G}^T$ may be written as:

$$\mathbf{G} = \begin{bmatrix} \mathbf{r}_0 \\ \mathbf{r}_1 \\ \mathbf{r}_2 \\ \vdots \\ \mathbf{r}_{N_2-1} \end{bmatrix} \quad and \quad \mathbf{G}^T = \begin{bmatrix} \mathbf{r}_0^T & \mathbf{r}_1^T & \mathbf{r}_2^T & \dots & \mathbf{r}_{N_2-1}^T \end{bmatrix}$$

and therefore

$$\mathbf{G}^T\mathbf{G} = \left[\mathbf{r}_0^T\mathbf{r}_0 + \mathbf{r}_1^T\mathbf{r}_1 + \ldots + \mathbf{r}_{N_2-1}^T\mathbf{r}_{N_2-1}\right] \tag{2.5}$$

Each term $\mathbf{r}_i^T\mathbf{r}_i$ is a full matrix of rank one.

By comparison, the covariance matrix update for RLS is defined as:

$$\mathbf{P}^{-1}(t) = \mathbf{P}^{-1}(t-1) + \phi(t)\phi^T(t)$$

The calculation of $\left(\mathbf{G}^T\mathbf{G} + \lambda\mathbf{Q_n}\right)^{-1}$ can then be performed as $(N_2 - N_1 + 1)$ updates to a matrix, initially equal to $(\lambda\mathbf{Q_n})^{-1}$. If the matrix is stored in a UDU factored form, then the same subroutine may be used for both this and the covariance matrix update, thus minimizing the programming required. Of course, $(\lambda\mathbf{Q_n})^{-1}$ must be available at the start of each time step, but it may be calculated ahead of time and changed only when the control weights are altered. In the actual implementation in RTGPC, the diagonal factor of $\mathbf{Q_n}^{-1}$ is divided by λ each time step, since λ is expected to be a primary tuning parameter, and may therefore change often.

Feedforward Control

Provision for feedforward control is made in RTGPC through the inclusion of auxiliary model inputs. Whether these feedforward variables are process inputs or outputs is irrelevant; if there is any correlation between them and the controlled process output it is identified by the process estimation system and then used automatically in the GPC prediction. Only in a few special circumstances does GPC perform complete dynamic feedforward disturbance rejection. Normally, such aggressive control

is either numerically unsound or impossible. There has been research into the separation of feedforward and feedback in the cost function (e.g. Tuffs, 1985, Mohtadi, 1986, M'Saad *et al.*, 1987) but they result in another controller tuning parameter which must be chosen by the user. Feedforward and feedback are left together in RTGPC and the controller is given the authority to make its own decisions regarding the feedforward control action.

2.2.4 Database Management

The on-line GPC program and the user interface communicate via a shared database file. The database file contains a maximum of 15 records, each approximately 13400 bytes long, indexed by the tagname. Each time step RTGPC reads and writes the record associated with the tag of the loop being controlled. (At present, RTGPC can only control one loop per image, but this limitation is just from the absence of multi-loop timing logic.) To optimize the speed of execution of RTGPC, it is necessary to minimize both I/O and the size of the executable image. Only one read and one write operation are permitted to the database file per time step, and so all the information associated with a controlled loop is contained in a single record. There is another database file, which contains the names of various tags and variables. It is accessed only by the user interface, GPC.EXE. The second file is necessary because VMS limits the size of a single record to 4095 longwords, or 16380 bytes, which imposed limits on the amount of information to be stored in the original database file.

The default VMS disk buffer is the same size as the record, and the same buffer is used for reads and writes, so in normal operation RTGPC need never wait for disk access, as the record will be already in the buffer. This of course is an additional advantage of reading and writing just one record, as multiple records could not all fit in the buffer.

The T filter is an infinite impulse response (IIR) filter. As such, the filter

output depends on all previous inputs. If only the old unfiltered variables (inputs) are to be used to calculate the new filtered value, then an infinite number of old measurements must be retained. Instead, old filtered values are used, thus reducing the amount of stored past data to finite, manageable quantities.

The data (both filtered and unfiltered) are stored in a fixed-length ring buffer. Each time step the oldest existing measurement is overwritten with the newest. The position of the newest measurement therefore advances one element in the array each time step, until it reaches the end of the array, whence it jumps to the first element. Keeping track of the positions of the newest, second newest, third newest, etc. measurements is an important task, maintained by a one dimensional array, the index. Index(1) is a pointer to the newest element in the array, index(2) points to the second newest, and so on. This form of indirect addressing permits the filtering routine to be ignorant of the state of the ring buffer and thus tidies up the calculations.

2.3 User Interface

The user interface is a standalone program which presents to the user a number of screens, each of which is used for a different purpose. For instance, some of the screens are used to specify the process model or controller configuration, while others are available for on-line diagnostics or overviews of a number of operating loops. Roughly speaking, the program can be divided into two parts, the specification (or input) and the diagnostic (or output) sections. The specification part of the program consists of a number of screens arranged in logical order, and some input is required of the user in many. The user may proceed from page to page in the prearranged order to enter the information for a new loop, or may move directly to the appropriate page when tuning an operating controller. The diagnostic section is not sequential because of the nature of diagnostics: the user may not be concerned at all about some parts of the operation and usually wants to examine only one particular aspect

of the operation.

Program design was complicated by the requirement for *all* the parameters of the adaptive controller to be accessible for change by the user. In the end, the difficulty of safely changing the model order on-line imposed the restriction that the model order and dead time must stay the same for a given run. The user may change the values of all other parameters and tuning constants while the adaptive controller is operating.

It is quite easy to move from one section of the program to the other, and although some of the commands differ between the two sections it is difficult to become confused because there are so few commands or opportunities for user input in the diagnostic section. For example, there is no way for the operator to alter any of the controller settings within the diagnostic side; the operator must first move back to the input section. This may seem inconvenient, but it is achieved with only a few keystrokes and facilitates a logical division of the program into two semi-independent halves which can be learned separately. There are two screens which are exceptions to this division: the title page (or main menu) and the header. The title page serves as a gateway between the sides and the header identifies the control loop in question regardless of the operator's position in the program.

A full description of the user interface is given in Chapter 3 of *The Adaptive GPC User's Guide*. The model structure screen is shown as an example in Figure 2.2. The text reminds the user of the meanings of the different variables to be specified. The default values are shown, corresponding to a first order model with a zero, and a first order T filter. A zero is used because of the likelihood of a fractional dead time: if the actual process dead time is not an integer multiple of the sample time, then the process has a fractional dead time which must be represented using a zero (using modified z-transforms). The delay to be entered by the user is the physical or continuous-time delay, exclusive of the unit delay arising from use of a zero order hold element. The section on this screen in the user's guide reads in part:

This is the first form after the header. It is reached by the Forward (Gold F) or Specify Model (Gold M) command. It is used to specify the structure of the process model used by the GPC controller. Only the sizes of the different polynomials are chosen using this screen, not the actual values of the coefficients. Note that there are six polynomials and only four of them can have delays. These four correspond to the four possible model inputs: the control action and up to three feedforward (measured disturbance) variables. The other two polynomials are the transfer function denominator, which cannot have a delay, and the T filter, which is used to filter the process inputs and outputs, and so cannot have a delay either. In this form [screen], the user chooses the orders of the six model polynomials. The order is defined as the highest power of z^{-1} present in the polynomial.

PLANT AND DISTURBANCE MODEL STRUCTURE

```
Plant output    Plant input    Disturbances          Unmeasured Dist.
                                                     and Noise
  A(z)*y(t) = B(z)*u(t-d) + D1(z)*v1(t-vd1) + T(z)*??
                          + D2(z)*v2(t-vd2)
                          + D3(z)*v3(t-vd3)
```

POLYNOMIAL	ORDER	DELAY
A(z)	1	
B(z)	2	0
D1(z)	0	0
D2(z)	0	0
D3(z)	0	0
T(z)	1	

Figure 2.2: Model Structure Page

Once the user has entered all the desired values and requested to move to another page, the program checks that all values are positive, within the permitted maximum values, and that the orders of A, B, and T are all nonzero. Only then are the checked data added to the database record for further use. If there is something wrong with one of the values then the user is informed and returned to the screen.

There are another eight specification screens:

- the $P(q^{-1})$, $Q(q^{-1})$ and $R(q^{-1})$ weighting polynomials,
- N_1, N_2, NU and λ,
- identification tuning parameters: standard deviation of the noise, memory length, deadbands (minimum and maximum) and initial covariance matrix,
- four screens for specifying tagnames for variables,
- and the initial model parameter value specification.

All of the screens check that the user has specified a controller that may work (e.g. N_2 must be at least as big as N_1), but because the implementation is for research purposes there are few other restrictions.

The diagnostic side of the program is composed of six screens: two overviews and four showing specific details. The two overviews are the listing of the entire database showing the status of all existing loops and the main diagnostic screen which shows some details of a single operating loop. The four detail screens are:

- the controller performance display, which shows the recent control history: setpoint, controlled and manipulated variables, for the last 15 control intervals.
- the controller transfer function display, showing the actual coefficients of the controller polynomials. The feedforward polynomials are not shown due to a lack of space, but they are available from a printout of the database.

- the model coefficients display, with the present values of the model parameters.
- the prediction page. On this page are displayed the predicted plant outputs and control actions for the next few control intervals.

Any of the diagnostic pages may be reached from any other, and as well as returning to the overview to inspect other loops it is possible to move directly to the controller tuning page of the specification section, in order to retune the controller if necessary.

2.4 Testing and Verification

The adaptation and GPC calculation routines were tested at the University of Alberta before being exported to the Fort Saskatchewan site. The user interface was written at Dow, using Dow equipment, and so was debugged and checked there.

The answers from the adaptation and GPC calculations were compared to the answers given by previously debugged MATLAB code. MATLAB is a programming language of such power that the RLS calculations take fewer than ten lines. All matrix bounds are checked automatically, and the numerical robustness of the algorithms used (e.g. for matrix inversion) is excellent. For example, the numerical robustness of MATLAB makes UDU factorization of the covariance matrix superfluous for small matrices and finite runs. MATLAB was originally written by Cleve Moler, who also was involved in the LINPACK project — a set of very robust, computationally efficient FORTRAN programs for solution of systems of linear equations. Moreover, MATLAB programs sufficiently resemble equations as they are written in books for debugging often to be just a matter of checking equations.

Each time a subroutine was written, it was tested, first in isolation and then in conjunction with the rest of the program. Its answers were compared to those given by MATLAB, and the FORTRAN program was examined until the answers

always agreed. Occasional hand calculations were performed to check the MATLAB code. CodeView, the MicroSoft source level debugger, was used to examine the FORTRAN program during execution and confirm the proper filling of matrices, evolution of variables, and so on.

The user interface developed at the Fort Saskatchewan site contains few calculations but is much more involved in terms of control flow. Debugging was performed by examining the actions of each routine as it was added to the program. Testing of subroutines in isolation was difficult, because virtually all subroutines affect the database and need access to the file. All file access is provided through a single subroutine, so the minimum configuration that could be tested is about four routines. In addition, most of the trouble was over control flow, often caused by interactions among different routines and different flags.

The VAX/VMS full-screen debugger was used to examine the operating program, and in particular control flow and the database data structure. Because of the size of the program, and the use of a single "INCLUDE" file to specify the database structure, it was necessary to take extreme care over changes to the database. Several errors occurred from "old," apparently debugged subroutines using outdated database definitions.

In addition, the user interface was used for over fifty different experimental runs, some of which are reported in the following chapter. By the end of the experimental evaluation of the software the user interface had functioned flawlessly for two weeks of continuous use.

Chapter 3

Experimental Verification and Performance

The software described in the previous chapter was tested at the Fort Saskatchewan, Alberta site of Dow Chemical Canada Inc. The purpose of the testing was to ensure that the adaptive GPC program gave the correct controller parameters for a given set of input data. It was also necessary to ensure that the software would work in the presence of problems encountered in the real world, such as missing or faulty measurements, or floating point calculation errors. The user interface was also evaluated for ease of use and to ensure that user specifications were correctly acted upon.

The software was verified on a pilot-scale process, but nevertheless under conditions similar to normal industrial conditions. The process itself was a bench-scale heater, but all data acquisition equipment and computer hardware were identical to standard industrial process control equipment. As such, the hardware configuration was ideal for evaluating implementation issues. The environment exactly mirrored a real plant, but more difficult control configurations could be examined than would be permitted on an actual production unit.

Despite the fact that the main purpose of the testing was to evaluate the software rather than GPC as a control algorithm, some conclusions were drawn from

the experiments regarding appropriate choices of tuning parameters for estimation and control. Unfortunately, some of the issues raised during the experiments were site-specific and as such Dow Confidential. Those issues are discussed in the Dow Confidential supplement to Shook and Shah (1989), but cannot be addressed here. Other issues more generic to the implementation of adaptive controllers were raised, and are discussed throughout this thesis.

3.1 Equipment Description

A schematic diagram of the experimental equipment is shown in Figure 3.1. The heater is constructed of 0.0016 metre (1/16 inch) thick carbon steel, and is 0.76 metre high, 0.76 metre wide and 0.22 metre from front to back. The back panel is steel and the front panel is 0.0064 metre (1/4 inch) plexiglas. Two thermocouples measure the air temperature immediately after the two heating elements and another five are placed at intervals along the air flow. All seven thermocouples are mounted on probes and extend well into the air flow. The heaters and fans are all rated at 600 Watts.

The serpentine path of the air through the heater is caused by a set of four horizontal baffles in addition to the vertical divider separating the two blower/heater elements. It is the interaction of the air with the back panel and these baffles that gives the heater its long dominant time constant. The air flow rate is fixed at approximately 0.0033 cubic metres per second so that the transit time from the heater elements to thermocouple 6 is approximately 30 seconds. At these conditions the Reynolds number is approximately 1300 so flow is still laminar, or would be were the channel long enough for the flow to develop fully. The flow regime has a significant effect on the process dynamics, as will be seen later.

The temperatures at thermocouples 3 and 6 were used as controlled variables. Thermocouple 6 provided a more challenging control problem, but thermocouple 3

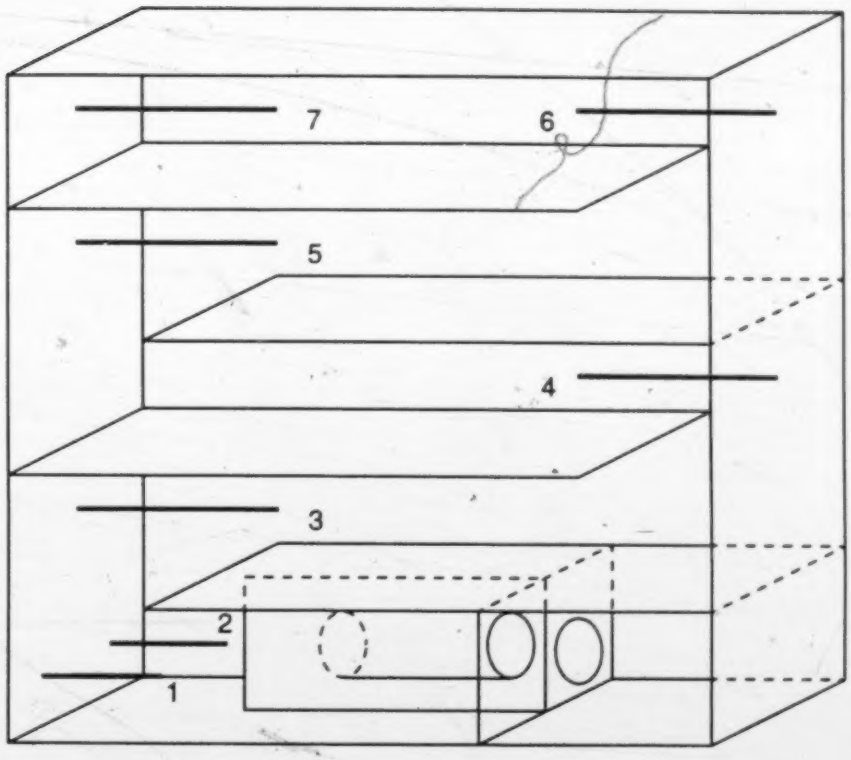

Figure 3.1: Schematic Diagram of Experimental Heater System

was more convenient for debugging purposes. The process dynamics between the heater elements and thermocouple 3 are much faster than for thermocouple 6, and process changes could therefore be made more frequently when thermocouple 3 was used.

The manipulated variable for all experiments was the power output from the rear heater. The front heater power output was the main measurable disturbance. The duty cycle was used to change the power output of both heaters. This way the control action effect on the plant (average power input) varied linearly with the controller manipulated variable (duty cycle) and the disturbance. If current had been used as the manipulated variable, then the control action effect would have varied with the square of the manipulated variable and the process gain would have varied enormously from 0 to 100% control action. The control configuration was designed and implemented by Dow personnel and is a good example of how to make the correct choice of manipulated variable.

Open Loop Behaviour

Typical open loop responses to control action and disturbance are shown in Figures 3.2 and 3.3. The sample time was 6 seconds, and the temperature was measured at thermocouple 3. Two things are immediately obvious: the low process gain and the high noise level. A 10% change in control action results in a steady state change in temperature of less than 2° C. The noise level meanwhile is at least 1/4° C. The magnitude of the noise was a consequence of using a long period duty cycle on the heaters in conjunction with a low air flow rate. The low Reynolds number flow restricts the amount of radial mixing. Under these conditions of high measurement noise a measurement filter is of prime importance to reduce the high frequency sensitivity of the controller.

Although the control and disturbance heater elements are exactly the same, they have significantly different effects on the process. This is because of the low

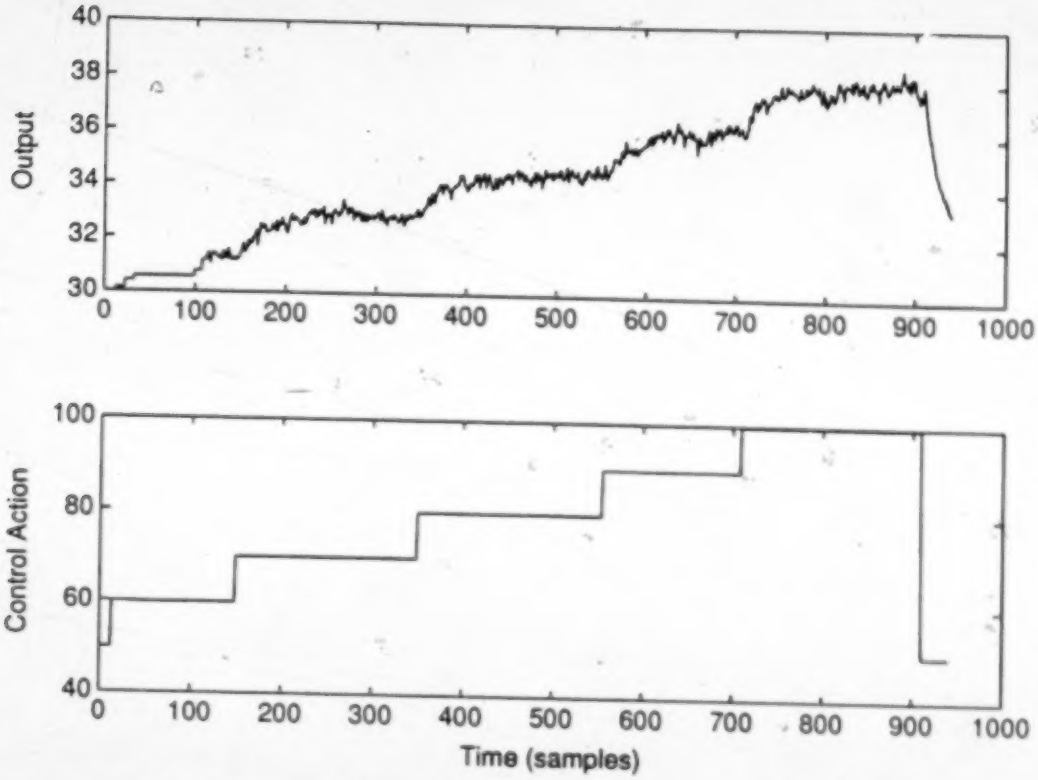

Figure 3.2: Experimental Open Loop Response to Control Action Change

flow rate: the air flow is not perfectly mixed. The air from the rear heater element (the final control element) heats the back panel directly while the front heater (the disturbance) heats the air on the front side. The front of the heater is covered with plexiglas, which is thicker and has a lower thermal conductivity than the sheet steel on the back. The extremely long response time for the disturbance step is a result of the asymmetric construction of the heater.

3.2 Experimental Performance of Software

The software was evaluated in an experimental environment to verify it under conditions matching actual industrial conditions as closely as possible. The software was intended to be capable of controlling, unsupervised, an operating production facility,

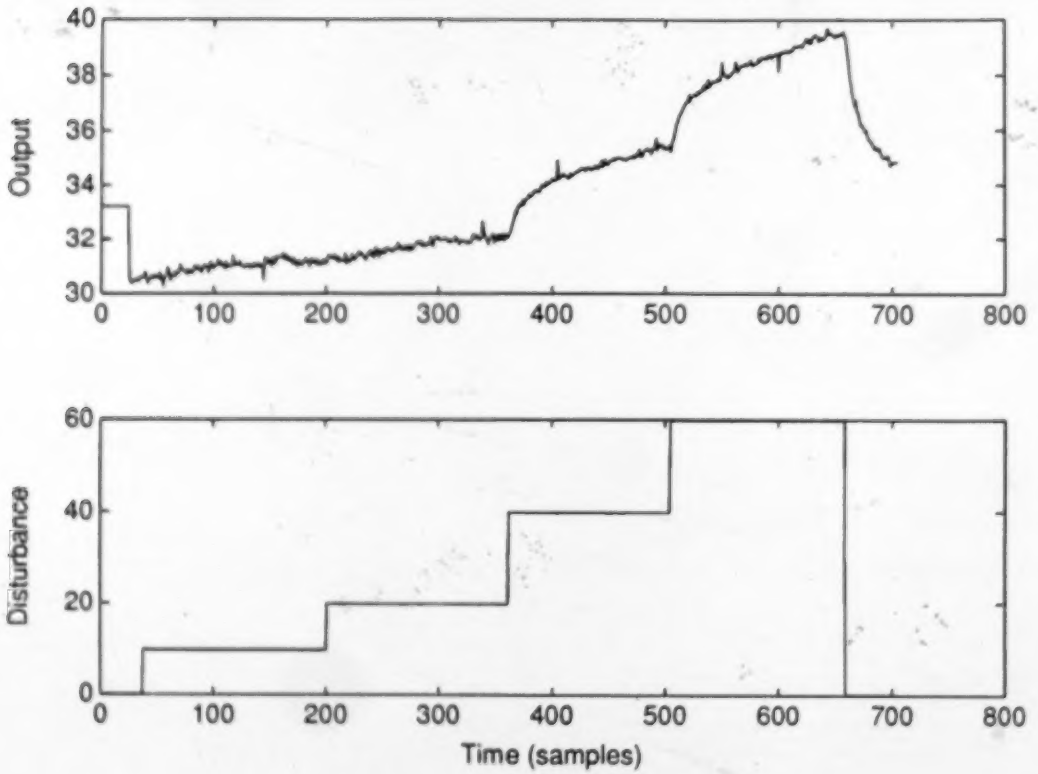

Figure 3.3: Experimental Open Loop Response to Disturbance Change

so experimental verification was a necessity. The three components of the control system: the user interface, the real-time executive and the DDC control program, were observed to evaluate their performance. The user interface was evaluated for its suitability for continuous on-line monitoring of the controller and the operating loop. The real-time executive was watched to ensure that the controller transfer function coefficient values were correct for the GPC and RLS tuning parameter choices, and the entire control computer system was checked to ensure that the final control action as implemented was correct for the parameter estimates and tuning.

Most of the tests consisted of comparing the results from the program with hand calculations. The values were printed out by the user interface, and were checked using the VMS on-line debugger. By the time the program was run on the experimental equipment, the portion of the program responsible for the actual adaptive GPC calculations had been tested in both simulation and experimental conditions at the University of Alberta and in simulation at Dow's Fort Saskatchewan site. The user and process interfaces were the parts of the software that were really being scrutinized for Dow Chemical Canada.

Some of the data from the experimental evaluation are reproduced in this chapter to illustrate some of the features of this particular adaptive implementation of GPC.

Consider the closed loop run shown in Figure 3.4. For this run the disturbance heater was off, thermocouple 3 was used as the measured variable and the sample time was 6 seconds. The "default" GPC tuning parameters were used for this run (see, e.g. McIntosh, 1988 and Shook and Shah, 1989). The run consisted of a pretuning period with the GPC controller in manual followed by transfer to automatic mode. During the large step change in setpoint the control action saturated for quite a long time, with no sign of reset windup. The step down was prompt and well-behaved. The high noise level could have been reduced somewhat through increasing λ, the control action weighting in the GPC cost function, but this was not done for this

run.

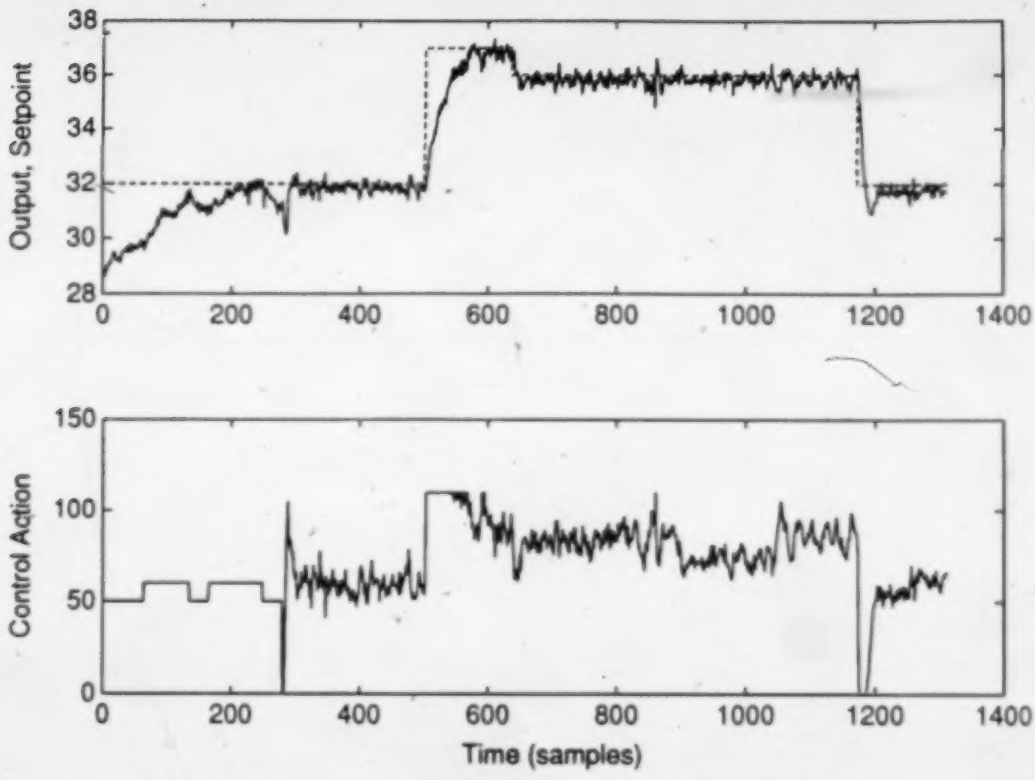

Figure 3.4: Closed Loop Performance of Adaptive GPC for Run 1

The door to the laboratory was opened after about 820 samples, and the surrounding air temperature rose by approximately 3° C. The effect of this disturbance is negligible in the temperature plot, but it may be seen clearly in the change in control action. When the door was closed and the air conditioning turned back on after 1020 samples, the controller responded well again.

Although the controller functioned well throughout the experiment, the same could not be said of the identification. The (filtered) prediction error is shown in Figure 3.5. The outliers at 859 and 860 samples are the result of a measurement error. The first outlier was caught by the prediction error maximum of 1.0 and so the model update was abandoned for that sample. The prediction error for the next

sample was slightly less than 1.0, so the second outlier was actually used for the model update. This is unfortunate, since it was 5 standard deviations away from zero and was therefore an outlier by any standard. The prediction error maximum should have been set to 0.5 instead of 1.0.

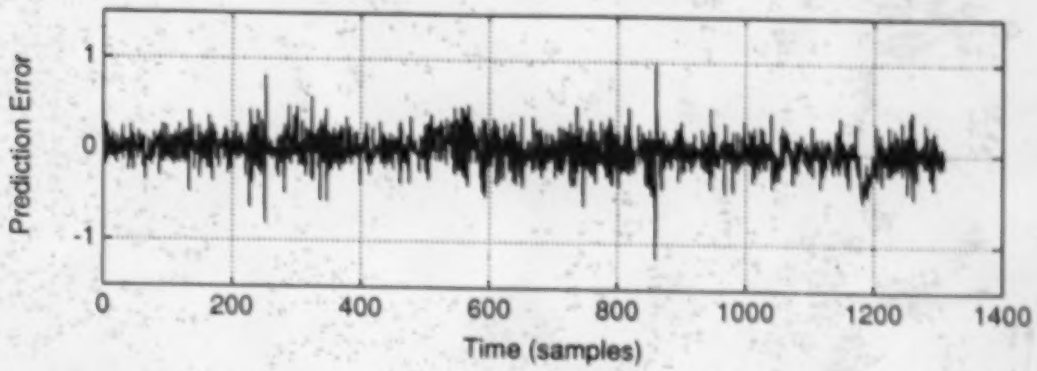

Figure 3.5: Prediction Errors for Run 1

The prediction error deadband for the run was 0.2°C. The standard deviation of the prediction errors was 0.18. Consequently, the prediction error deadband was used 998 times out of 1311 samples in the run, so only 313 data points were used to calculate the model parameters. Obviously a smaller deadband would have permitted the use of more measurements and given greater precision in the predictions. The minimum value of the forgetting factor was 0.993, which corresponds to a memory length of 140 measurements. The average memory length was considerably longer. The logical consequence of a long memory length is a slow rate of change of the parameters. Since the minimum memory length was 140 and only 313 measurements were used to calculate the model, it is not surprising that the controller performed unsatisfactorily. The large oscillations in control action and temperature between 1060 and 1160 samples are the result of poor parameters. Only after excitation was introduced during the final step change did the parameters reach more sensible values.

The *controller* transfer function parameters are shown in Figure 3.6. The sudden jump from zero is an artifact of the recording process: the controller polynomials simply do not exist until the controller is switched into automatic mode. They can be seen to change slowly over time and spend long periods of time without changing at all. The rate of change is limited by the high forgetting factor. Lower forgetting factor values would permit faster parameter changes, because the adaptive gain (the covariance matrix) would be allowed to grow much more quickly during times of poor prediction. The high forgetting factors are of course caused by poor tuning of the forgetting factor algorithm.

In contrast, the response shown in Figure 3.7 is not as noisy because of better adaptation. The GPC controller has the same default tuning, but the adaptation is much faster. The forgetting factor was typically lower for this run, resulting in a shorter memory length and faster adaptation. There is of course a greater risk of covariance blowup, but still not much, since a prediction error deadband is in use. The control signal is noisier over the last 70 or so samples because of the unmeasured disturbance: the air conditioning cut in again.

The controller parameters for Run 2 reflect the more rapid adaptation, as can be seen in Figure 3.8. The parameters adjust to reflect changes in the process. As for Run 1, only a first order process model was used. The difference in the adaptation between Runs 1 and 2 was a relatively simple change: the "asymptotic memory length," the primary tuning parameter for the identification, was 1000 for Run 1 and 200 for Run 2. The asymptotic memory length may be thought of as the number of measurements retained in the covariance matrix once the identification is at steady state. Obviously a high value will give slow adaptation but too low a value will result in parameters that change too quickly.

The controller transfer function depends on the process model and the controller tuning parameters. With this equipment, changes in ambient temperature and external heat transfer coefficient can cause the process model parameters to change

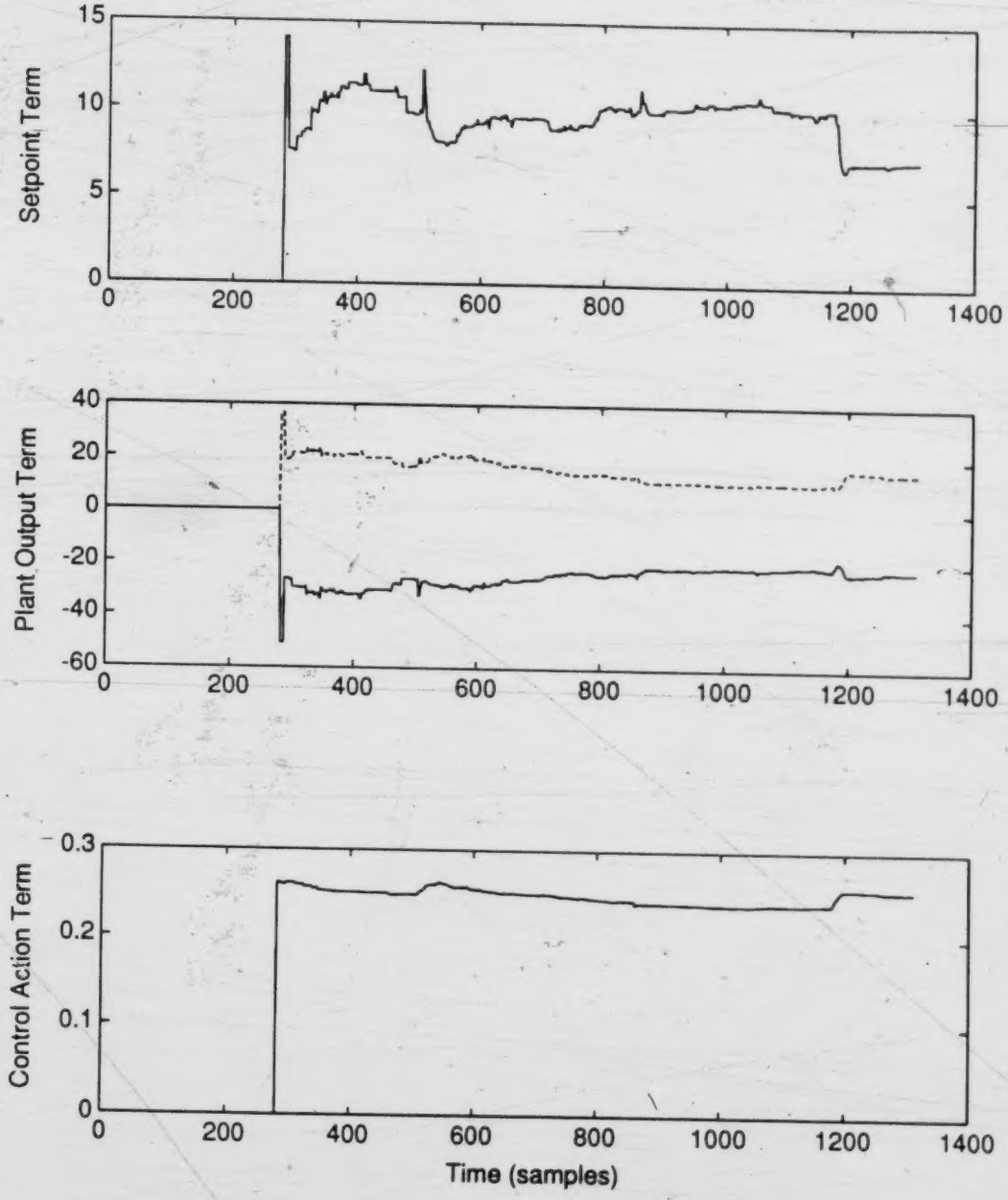

Figure 3.6: Controller Transfer Function Parameters for Run 1

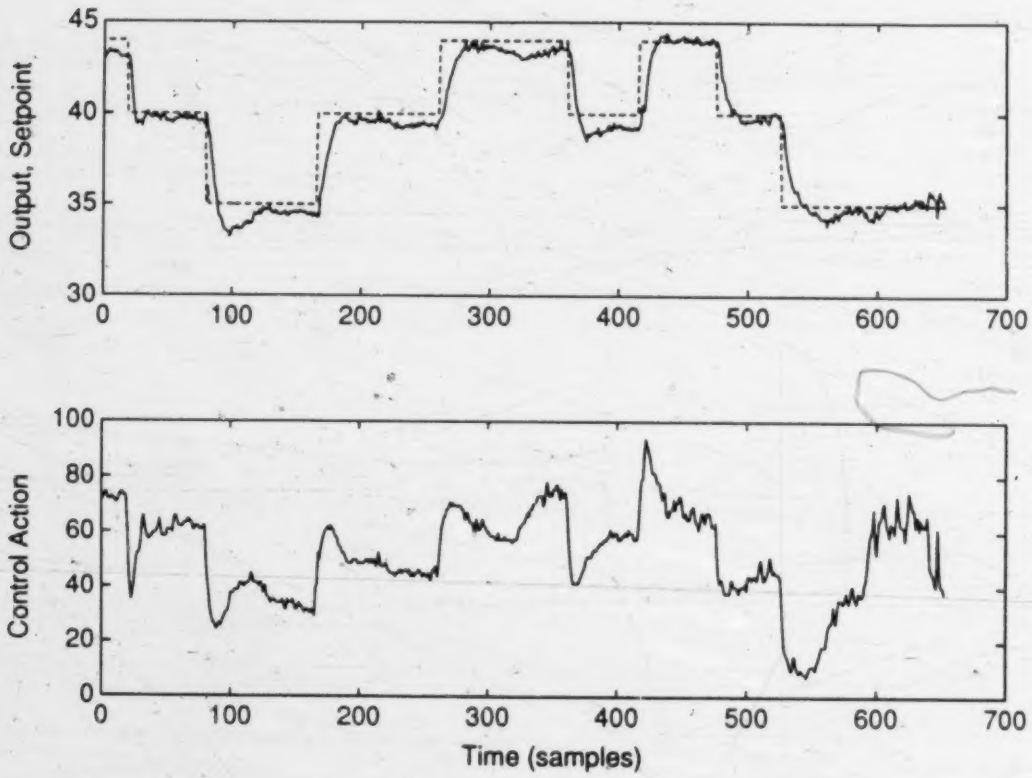

Figure 3.7: Closed Loop Performance of Adaptive GPC for Run 2

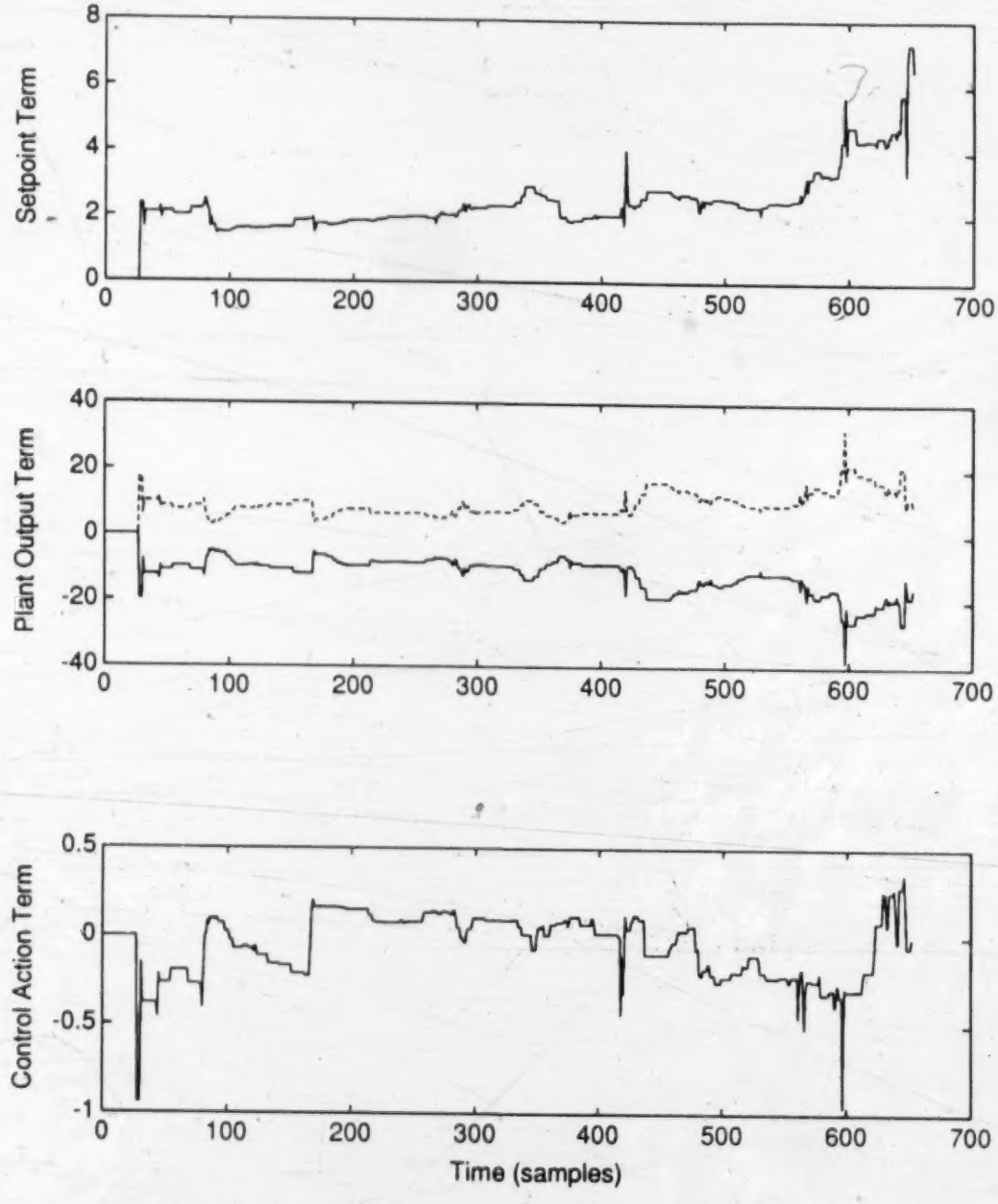

Figure 3.8: Controller Transfer Function Parameters for Run 2

even when the temperature is constant, as can be seen in the results of Run 3. The process inputs and outputs are shown in Figure 3.9 and the controller parameters are displayed in Figure 3.10. Even though the nominal process condition — the controlled variable — was well-controlled at one value, the control action required to maintain the temperature changed throughout the run.

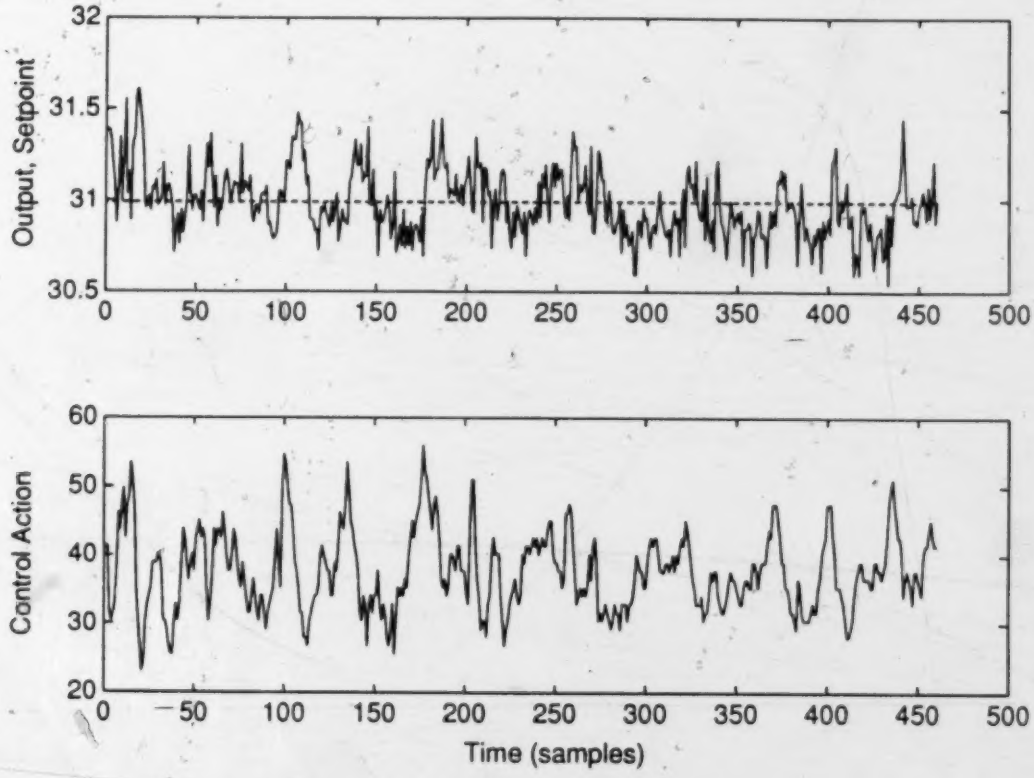

Figure 3.9: Closed Loop Performance of Adaptive GPC for Run 3

3.2.1 Operation in the Presence of Communication Difficulty

The software was required to work in the presence of operational difficulties, both sensor and communication failures. This requirement is the main difference between this implementation and the majority of pilot-scale adaptive control applications. It

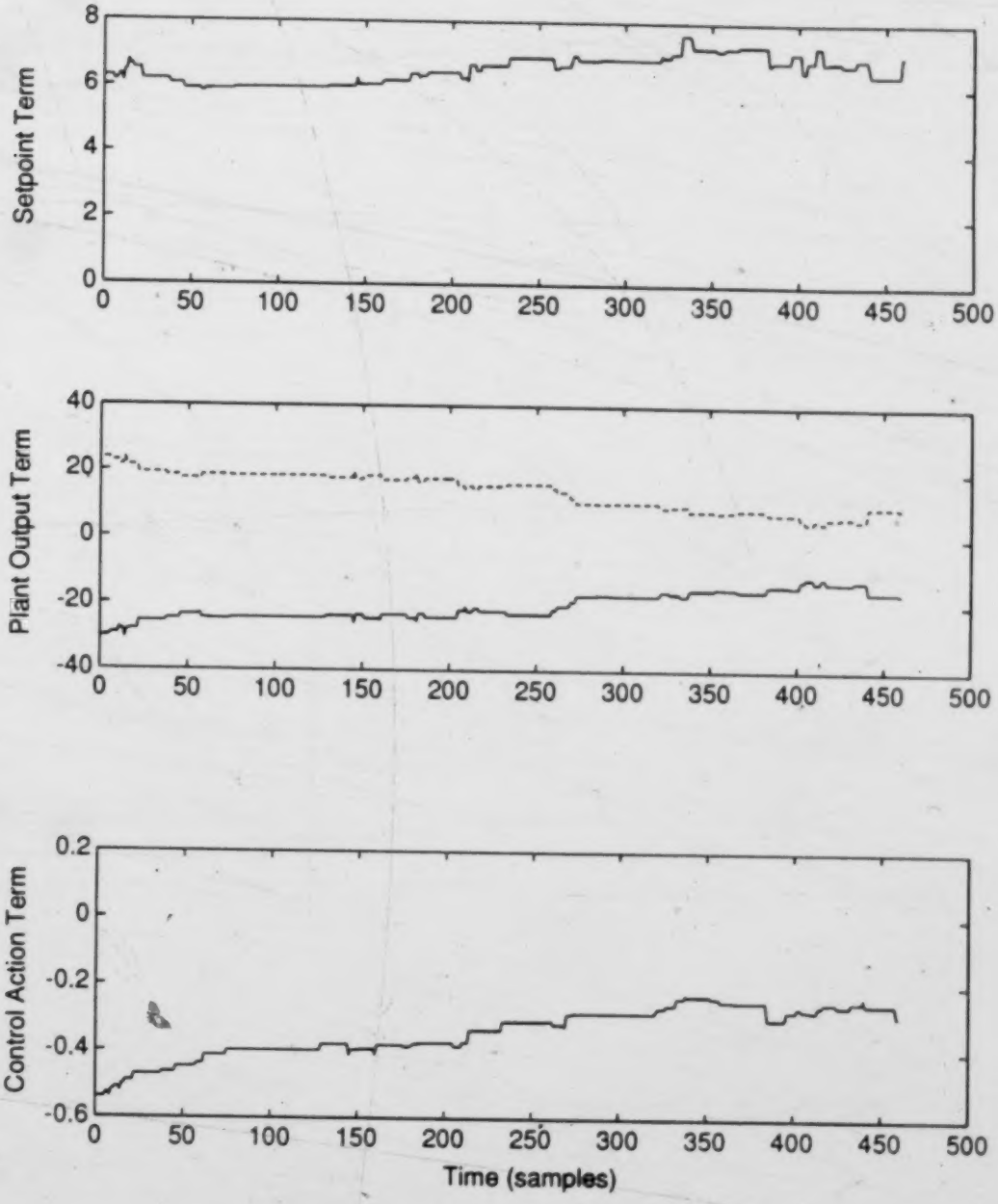

Figure 3.10: Controller Transfer Function Parameters for Run 3

is a reasonable requirement for any control system, and doubly so for an adaptive controller since bad measurements affect more than just the immediate control action: the process model may be corrupted.

The separation of the adaptive control effort into two computers was described in Chapter 2. The separation was done to ensure that the control action would be implemented regardless of the state of communication between the different components in the distributed control system. Thus control would be maintained in the event of a communication failure of any length. In practice communication problems were relatively common and the capacity of the program was tested sufficiently.

The steps taken to make the controller robust (to communication problems) were outlined in Chapter 2. If any single value was missing, it was replaced (for the purposes of filtering) by a substitute and the parameter update was abandoned. The substitute was usually the previous value of the missing variable, but for the process output the predicted value was used because it is the most accurate value available. For some early experiments, such as Run 1, even the process output was replaced with its previous value, since the parameter estimation system had yet to demonstrate itself as accurate in an on-line mode. The value was of course only used for filtering.

The communication problems experienced during Run 1 are shown with the process output in Figure 3.11. Each dot above the process output identifies a time step at which the temperature was missing. Altogether there were 38 missing measurements during 1300 samples, and all occurrences of missing data were handled well. The one serious measurement error (as opposed to communication error) occurred at the 859^{th} sample, immediately after a communication error at the 858^{th} sample. The response of the process to the measurement error was described above.

In short, the software performed adequately in the presence of real-time communication difficulties, although the prediction error maximum must be chosen carefully as too large a value will cause the behaviour seen in Run 1.

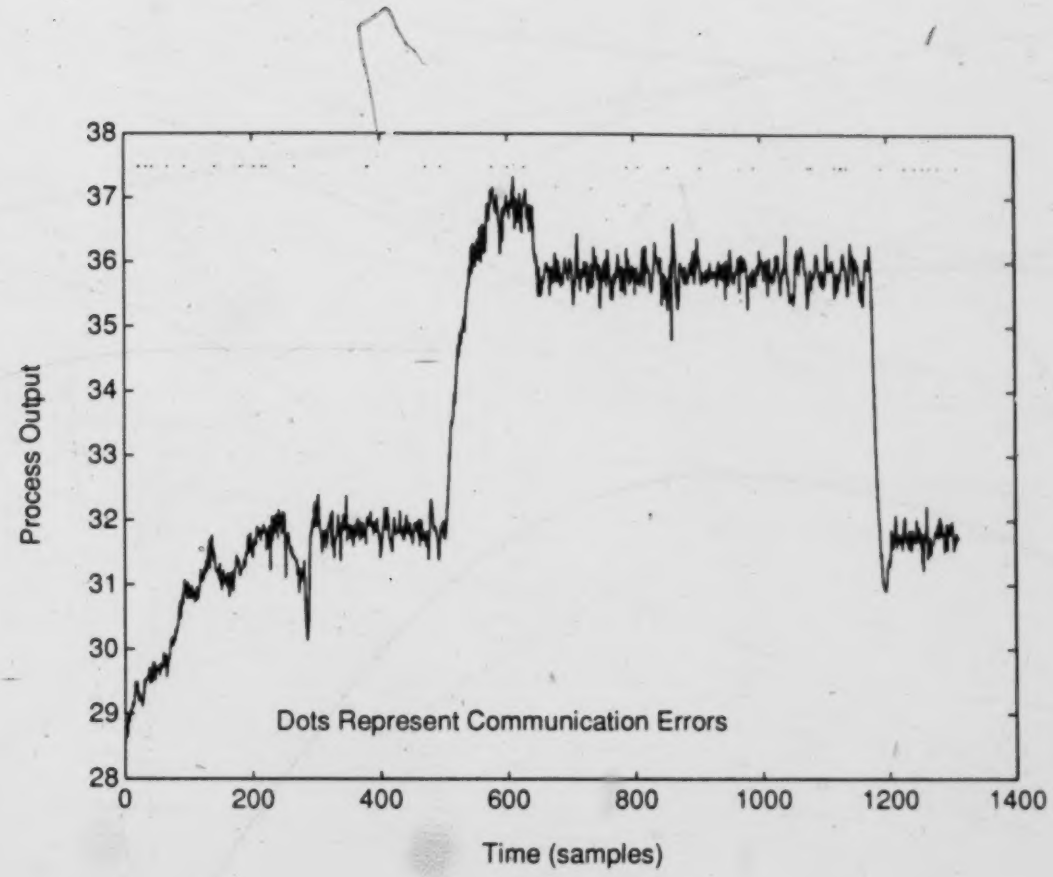

Figure 3.11: Process Output and Communication Errors

3.2.2 Feedforward Control Performance

One more requirement for the adaptive controller software was that it contain provision for feedback control in conjunction with feedforward (FB+FF), an example of Multi-Input, Single Output (MISO) control. In adaptive control implementations, feedforward control has two advantages. Firstly, it helps the controller catch disturbances before they can affect the process. This is an effect common to adaptive and fixed controllers. For adaptive controllers, however, there is an extra benefit: use of another measurement improves output prediction and reduces by one the sources of prediction error. Since the measurements used for feedforward control are usually the ones most responsible for process upsets, their consideration in the model can improve prediction immensely.

The process inputs and outputs for one FB+FF control run are shown in Figure 3.12. There were four changes in the disturbance during the run and the controller was able to compensate for the last two. This is not very clear from the process inputs and outputs because of the high noise level, but it is obvious from a plot of the prediction errors (Figure 3.13) that the model did not initially take the disturbance into account. The large prediction error at the 247^{th} sample is a result of the change in the disturbance two samples previously. Subsequent disturbances did not cause large prediction errors, despite the fact that they were the same size or larger than the one that caused the large prediction error.

3.3 Weaknesses of the Existing Algorithm

The adaptive control software demonstrated that it is capable of controlling a physical process under real operating conditions. It is a working implementation of a 1988 state of the art adaptive controller. Two concerns were raised during the experimental trials that could not be addressed within the time frame of the project. They were instead left until after the successful implementation because of their intricate

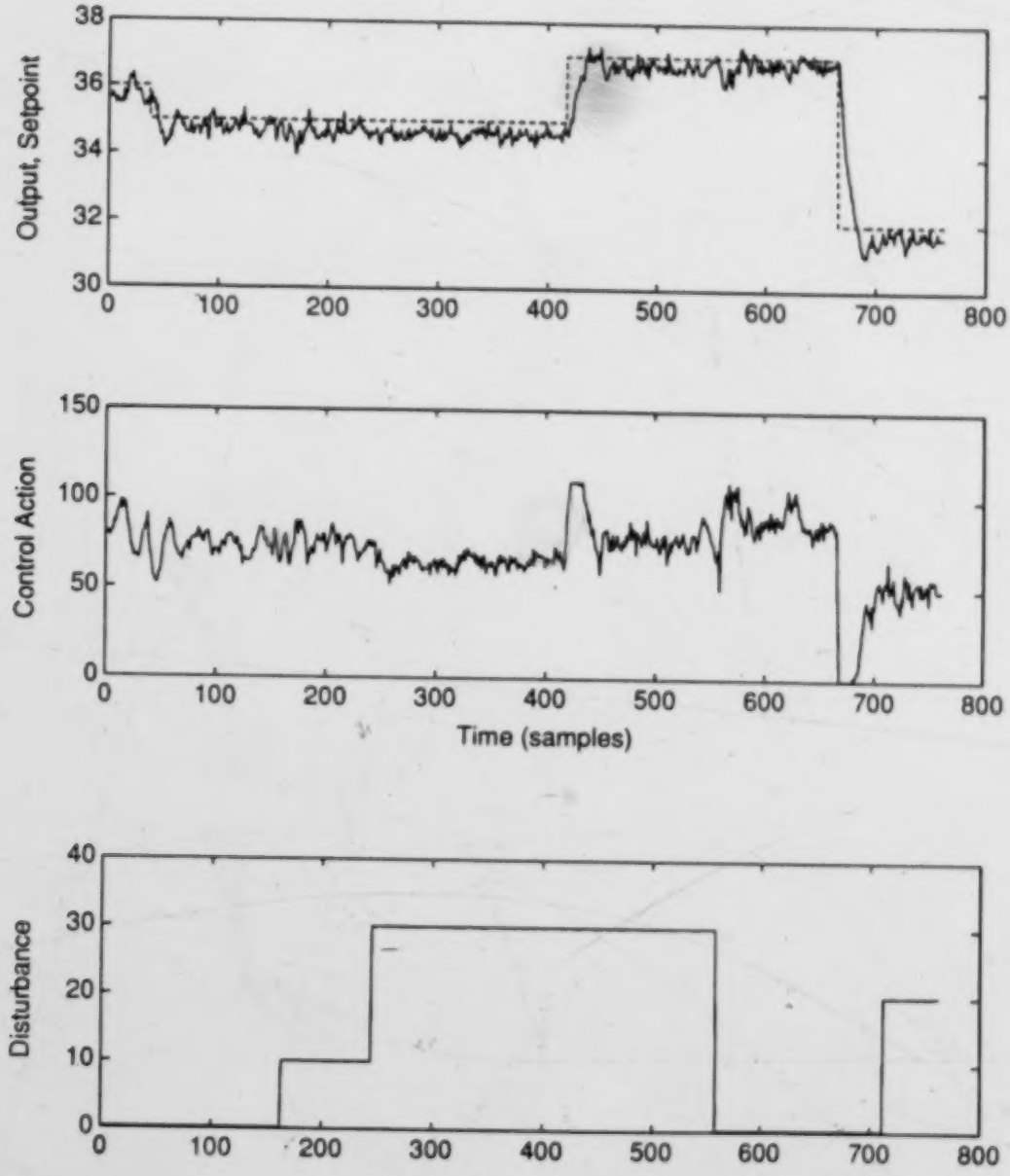

Figure 3.12: Closed Loop Performance for Run 4

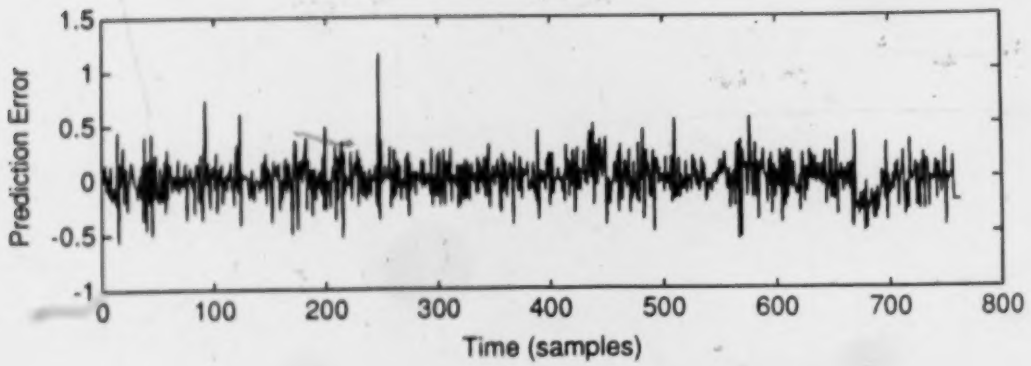

Figure 3.13: Prediction Errors for Run 4

nature.

Firstly, the controller uses a single T filter for both estimation and control. While this was the configuration originally supported by theory (Clarke *et al.*, 1987b), a single T filter is difficult to tune. There may be no single T which provides both sufficient robustness against model-plant mismatch and sufficiently active disturbance rejection. Instead, a number of researchers (e.g. E. Lambert, 1987, McIntosh *et al.*, 1990, Mohtadi, 1990, Wittenmark, 1990) found that two different T filters should be used: one for identification and a different one for control. The arguments were mainly heuristic but the experimental results were incontrovertible.

The use of a single T filter, while not crippling, does limit the performance that can be achieved by this adaptive control software. The whole topic of process identification for adaptive GPC was investigated to find the most relevant identification method. The investigation and results are discussed in Chapters 4, 5 and 6 of this thesis and selections have been published elsewhere (Shook *et al.*, 1989, 1990).

One other limitation to the adaptive control package is the use of an exponential forgetting factor, specifically the method of Ydstie *et al.* (1985). In conjunction with a prediction error deadband it has demonstrated good robustness properties, but careful tuning is sometimes necessary and in a FB+FF control context there is

potential for serious problems.

A prediction error deadband is necessary when Ydstie's forgetting factor is used. It must be chosen carefully; neither too small (or it will not work and there will be covariance blowup) nor too large (adaptation will be too slow). A way to calculate the noise variance on-line would be most useful for setting the adaptation rate appropriately.

Adaptive feedforward control is only useful if there is enough information in the measured disturbance for its effect to be identified. If, as was the case for these experiments, the disturbance occurs only rarely, it is possible for the effects of one disturbance to be "forgotten" before the next time. A *directional* forgetting factor would remove the possibility of such an event. The topic of data forgetting in the FB+FF control environment is discussed in Chapter 8.

Chapter 4

Identification for Adaptive GPC

The description of the adaptive GPC implementation in Chapter 3 emphasized the performance of the identification routine. The controller functioned well, except when the identification method failed to produce an acceptable model. The identification task performed poorly often enough to illustrate a well-recognized fact: the identification of the model is the weak link in adaptive GPC implementations (Mohtadi, 1990). The GPC control algorithm is less sensitive to model-plant mismatch than previous self-tuning controllers, but an accurate model is still needed for high-performance control (McIntosh, 1988). Two major concerns were raised in Chapter 3 regarding identification for adaptive GPC: (1) the model parameters must be updated appropriately, often a more complicated task in the MISO control case, and (2) the identified model is judged on the resulting control quality, so the identification task should be designed with this in mind.

This chapter provides an introduction to process identification methods for adaptive control. Although the method of recursive least squares is the most popular identification method used with GPC, this is largely because of historical precedent: RLS was used in the original GPC papers (Clarke *et al.*, 1987a,b), and in other, earlier self-tuning controllers (e.g. the Self Tuning Regulator, or STR, of Åström and Wittenmark, 1973). There are many methods to calculate the "best" model,

and in fact there are many ways to *specify* the "best" model. It will be shown later in this chapter that RLS is not the most appropriate identification method for GPC.

Three major parameter estimation methods are discussed in Section 4.1. All three methods provide solutions to the identification of time-varying processes. The intention is not to provide a rigorous theoretical analysis, but to describe the qualitative differences among the different approaches. Rigorous analyses have been performed by many others, e.g. Ljung and Söderström (1983) and Goodwin and Sin (1984) and the discussion here borrows a great deal from their work. The methods discussed are the Kalman filter, RLS and the method of Least Mean Squares (LMS) of Widrow and Stearns (1985).

While they have very different properties, all three methods represent perfectly valid approaches to the parameter estimation problem. The Kalman filter (Kalman, 1960) is the optimal solution to the parameter estimation problem under a certain fixed set of assumptions regarding the process and the measurement noise. RLS is not optimal, but is more flexible and can be modified in a large number of ways to fit different assumptions regarding the system. The third method, LMS, is computationally very simple and has some very attractive robustness properties.

More basic to the field of adaptive control is the question of identification objectives: how is the best process model defined? Before any meaningful comparison can be drawn among different methods, the goal of identification must be defined explicitly. For an adaptive controller, the "best" model is that which provides the best actual control quality. The original STR of Åström and Wittenmark (1973) was derived using an overall control objective. The method of least squares was chosen as the parameter estimation method because it was a practical solution to the resulting control-relevant identification problem. The use of an overall control criterion is revived in this chapter and applied to the Long Range Predictive Control (LRPC) problem. The "optimal" or control-relevant identification goal for GPC is derived. A resulting control-relevant identification method for adaptive GPC, Long

Range Predictive Identification (LRPI), will then be developed in Chapter 5.

4.1 Parameter Estimation Methods

The parameter estimation problem may be stated as follows:

> Given a process, a set of measurements and a model structure, determine the set of parameters such that the model best describes the behaviour of the process.

As was mentioned in Chapter 2, an ARIMAX (AutoRegressive Integrated Moving Average with eXogenous input) model is used for GPC. For the purposes of this discussion a simpler ARX (AutoRegressive with eXogenous input) model will be used:

$$\hat{A}(q^{-1})y(t) = \hat{B}(q^{-1})u(t-1-d) + \xi(t) \tag{4.1}$$

The equation can be rewritten in one-step ahead predictor form:

$$y(t) = [1 - \hat{A}(q^{-1})]q^1 y(t-1) + \hat{B}(q^{-1})u(t-1-d) + \xi(t) \tag{4.2}$$

or

$$y(t) = \phi^T \hat{\theta} \tag{4.3}$$

where the parameter estimate vector $\hat{\theta}$ and observation (or regressor) vectors are:

$$\hat{\theta} = [\hat{a}_1, \ldots, \hat{a}_{na}, \hat{b}_1, \ldots, \hat{b}_{nb}]^T \tag{4.4}$$

$$\phi(t) = [-y(t-1), \ldots, -y(t-na), u(t-d-1), \ldots, u(t-d-nb)]^T \tag{4.5}$$

and d is the process delay.

The "best," or "limiting" parameter vector is that which actually does provide the best match between the predicted and actual process output. It is designated θ^0. The plant itself may be linear or nonlinear, but it is not assumed that θ^0 can *completely* describe the process. In other words, θ^0 is not necessarily equal to the

true parameter vector θ^* of a linear finite-dimensional plant. Instead, θ^0 is the set of parameters which come closest to describing the process behaviour. The parameter identification problem is thus one of finding θ^0.

The acknowledgement that θ^0 cannot completely describe the process means that there will always be some part of the process output that cannot be predicted by the model. This residual prediction error is caused by measurement noise, disturbance effects, or the fact that the model structure may be too simple for the process (underparameterization).

Since θ^0 is the "best fit" set of parameters, it is necessary to define the "best fit." The vast majority of identification methods — but by no means all — minimize the square of the one step ahead prediction error. In other words, the "best" process description is that which gives the minimum variance one step ahead prediction error: $\mathbf{E}((y(t+1) - \hat{y}(t+1|t))^2)$.

This objective is chosen for two reasons: (1) the minimization is relatively simple for this kind of objective function, and (2) it is easy to analyze the resulting model and prediction quality. The one step ahead prediction error is linear in the model parameters, so the minimization is just a linear regression. Since a squared cost functional is used, there is a closed form solution to the linear regression for any given model order and set of data. In addition, it is simple to find the variances of the model parameters and the prediction error. There are identification methods which use other criteria, such as least absolute value of the prediction error, but they are more complicated and can be shown to be less relevant for GPC. The use of longer-range prediction errors will be addressed in Section 4.2.

The relationship between the process output and the limiting model can be defined by the following relation:

$$y(t) = \phi(t-1)^T \theta^0(t) + e(t) \qquad (4.6)$$

where $e(t)$ is the residual error: that part of the process response that cannot be predicted by the model. In contrast, the difference between the actual process output and that predicted by the current model (the estimate of θ^0, designated $\hat{\theta}$) is:

$$y(t) = \phi(t-1)^T \hat{\theta}(t-1) + \epsilon(t) \tag{4.7}$$

where $\epsilon(t)$ is the prediction error.

Since adaptive control is being used, the process is assumed to change over time. θ^0 will therefore change with time:

$$\theta^0(t) = \theta^0(t-1) + w(t) \tag{4.8}$$

The value of $w(t)$ (a vector) represents the change in the parameter vector between samples. If the form of $w(t)$ is known, then the optimal identification scheme can be defined. To take the trivial case, if $w(t) = 0$ for all t, then the best parameters do not change ($\theta^0(t) = \theta^0(0)$). The best least squares method would therefore have *no* forgetting (other than to remove the effects of initial conditions).

More often, the ideal model parameters are considered to change in some manner. Typically $w(t)$ is considered to be a vector of uncorrelated random variables. The parameters will then vary in a random walk or Brownian motion. If $w(t)$ is Gaussian and the covariance is known, then enough is known about the process to make a Kalman filter the best identification method (Kalman, 1960).

In chemical engineering applications, the process is often nonlinear or of very high order. In such cases even though the process remains the same, the best fit low-order linear model may change in different ways. If the form of $w(t)$ is known, then it is often possible to design an "optimal" identification method. More often, however, the form is unknown, so optimality is difficult to achieve. For example, there may be a slow drift of parameters as equipment ages. The model may have to change abruptly to meet a new operating condition following a setpoint or disturbance change. Variations in excitation may result in variations in the model parameters for undermodelled systems.

Often a change in setpoint is enough to force a change in the process model parameters. If this is the only way the parameters change then $w(t)$ will take the following form:

$$w(t) = \begin{cases} 0 & \text{with probability} \quad 1-\gamma^2 \\ v & \text{with probability} \quad \gamma^2 \end{cases} \tag{4.9}$$

where v is a random variable with some (unknown) distribution. If enough is known about the process, then the magnitude of v may be inferred from the amount of change in setpoint.

Detecting the times of transition (times when $w(t)$ is nonzero) is difficult in the general case, but in process control applications it is often possible to tell when a disturbance has entered the system or at the very least when a setpoint change has been made. A parameter estimation method has been derived for this kind of model change, but it does not seem to have been embraced by the process control field. Andersson (1985) formulated a scheme called AFMM (adaptive forgetting through multiple models). It is a highly complex method where several models are present. The most likely model is used, and the least likely one is discarded and the most likely model gives birth to a new one. It was not investigated thoroughly for this work but it is a promising approach.

Although there is a need for identification schemes that can cope with such parameter changes, there are relatively few schemes that have been formulated to handle such cases. Instead, most work has focussed on the simpler random walk case.

If the parameter variation is a random walk as described previously, and if the noise $\{e(t)\}$ is white Gaussian with variance $R_2(t)$, then it has been shown (e.g. by Ljung and Söderström, 1983) that the optimal estimate $\hat{\theta}(t)$ (which minimizes the least squares prediction error criterion) is given by the Kalman filter:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \mathbf{L}(t)\epsilon(t) \tag{4.10}$$

$$\epsilon(t) = y(t) - \phi(t)^T\hat{\theta}(t-1) \tag{4.11}$$

where the gain vector $\mathbf{L}(t)$ is given by

$$\mathbf{L}(t) = \frac{\mathbf{P}(t-1)\phi(t)}{\hat{R}_2(t) + \phi(t)^T\mathbf{P}(t-1)\phi(t)} \tag{4.12}$$

and the matrix $\mathbf{P}(t)$ is updated according to

$$\mathbf{P}(t) = \mathbf{P}(t-1) - \frac{\mathbf{P}(t-1)\phi(t)\phi(t)^T\mathbf{P}(t-1)}{\hat{R}_2(t) + \phi(t)^T\mathbf{P}(t-1)\phi(t)} + \hat{\mathbf{R}}_1(t) \tag{4.13}$$

when the estimated variances $\hat{\mathbf{R}}_1(t)$ and $\hat{R}_2(t)$ are equal to the actual variances $\mathbf{R}_1(t)$ and $R_2(t)$.

RLS with an exponential forgetting factor uses the same parameter update as the Kalman filter (equation 4.10) but the vector gain $\mathbf{L}(t)$ and the matrix $\mathbf{P}$ are calculated differently:

$$\mathbf{L}(t) = \frac{\mathbf{P}(t-1)\phi(t)}{\lambda(t) + \phi(t)^T\mathbf{P}(t-1)\phi(t)} \tag{4.14}$$

$$\mathbf{P}(t) = \mathbf{P}(t-1) - \frac{\mathbf{P}(t-1)\phi(t)\phi(t)^T\mathbf{P}(t-1)}{\lambda(t) + \phi(t)^T\mathbf{P}(t-1)\phi(t)} \tag{4.15}$$

This can be construed to be a special case of the Kalman filter, with $\hat{\mathbf{R}}_1(t)$ and $\hat{R}_2(t)$ chosen as follows:

$$\begin{aligned}\hat{\mathbf{R}}_1(t) &= \left(\frac{1-\lambda(t)}{\lambda(t)}\right)\mathbf{P}(t) \\ &= \left(\frac{1-\lambda(t)}{\lambda(t)}\right)\left[\mathbf{P}(t-1) - \frac{\mathbf{P}(t-1)\phi(t)\phi(t)^T\mathbf{P}(t-1)}{\lambda(t) + \phi(t)^T\mathbf{P}(t-1)\phi(t)}\right] \qquad (4.16)\end{aligned}$$

$$\hat{R}_2(t) = \lambda(t) \tag{4.17}$$

Variable forgetting factor methods (in particular the method of Ydstie *et al.* (1985)) therefore adjust the magnitude of $\hat{\mathbf{R}}_1(t)$ in an attempt to match $\mathbf{R}_1(t)$ in magnitude, without accounting for direction.

The LMS method of Widrow and Stearns (1985) is similar to RLS, but is much simpler. The gain vector $\mathbf{L}(t)$ is merely $\mu\phi(t)$. The normalized variant is only slightly more complicated:

$$\mathbf{L}(t) = \frac{\mu\phi(t)}{1 + \mu\phi(t)^T\phi(t)} \tag{4.18}$$

This once again corresponds to a special case of equations 4.12 and 4.13 with

$$\hat{\mathbf{R}}_1(t) = \mu^2 \frac{\phi(t)\phi(t)^T}{1+\mu\phi(t)^T\phi(t)} \tag{4.19}$$

$$\hat{R}_2 = 1 \tag{4.20}$$

$$\mathbf{P}(0) = \mu\mathbf{I} \tag{4.21}$$

μ is a constant and is the only user-specified parameter. It is typically on the order of 10^{-3}.

The initial condition for $\mathbf{P}$ in LMS is significant because given the standard update of equation 4.13, it can be seen that $\mathbf{P}$ does not evolve at all. The reduction in $\mathbf{P}$ from the update (the second term in equation 4.13) is exactly offset by addition of $\hat{\mathbf{R}}_1$. This algorithm therefore avoids both covariance windup and turnoff. LMS is however typically much slower in responding to an abrupt change in the true system because of the fixed, low adaptive gain. RLS in contrast can adapt much more quickly through forgetting. Forgetting allows the covariance matrix $\mathbf{P}$ to grow and thus increase the gain and therefore the rate of update of the parameters.

The RLS method has been used for process identification for GPC because it is easier to implement than the Kalman filter and yet more flexible than LMS. In addition, to implement a Kalman filter, values are needed for $\hat{\mathbf{R}}_1$ and $\hat{R}_2$. It is not always possible to find the right values, and using the wrong values destroys the optimal qualities of the method. The choice of parameter estimation method is still a difficult task for the designer. It is necessary to balance performance with computational requirements and speed of convergence with robustness. The use of prior knowledge of the process is also important for selecting the parameter variation model. As ever, an inappropriate choice will result in poor control.

All three of these methods have one thing in common: their objective. They are simply three different attempts to solve the same problem: getting good one-step ahead predictions. In cases where there is enough statistical information an optimal method similar to the Kalman filter should be used, in other cases RLS is more

appropriate and in still other cases LMS may be the best method. Nevertheless, all three methods are nothing more than variations on a theme. Whether or not the theme is appropriate for adaptive GPC is another question altogether.

4.2 Overall Control Criterion

The objectives of process control are regulation and setpoint tracking. Some quantitative measure of control quality is needed for any mathematical treatment. Many controllers measure the quality of control in terms of the variance of the control error, $(y_{sp} - y(t))$. This provides a reasonable performance function which penalizes large excursions from the setpoint and is easy to analyze.

The original STR of Åström and Wittenmark (1973) was an early attempt at minimum variance control. The objective of the STR was to minimize the variance of the actual control error:

$$\sigma_e^2 = \mathbf{E}\left(e^2\right) = \mathbf{E}\left((y_{sp}(t+1) - y(t+1))^2\right)$$

The STR was not particularly successful because it could not withstand significant amounts of measurement noise or model-plant mismatch. This drawback was caused by the simplistic design of the STR, not the overall objective. Specifically, only the one-step-ahead error was considered. This yielded a completely specified problem (finding one control action to force one output to the setpoint). Unfortunately processes exhibiting inverse reseponse could not be controlled at all by the STR. High frequency disturbances also tended to be amplified. More "cautious" or "robust" versions of the STR were developed, for example the GMV controller of Clarke and Gawthrop (1979) which used a variable control action weighting parameter to detune the control action and stabilize the closed loop.

GPC is more successful than the STR or GMV controllers because GPC considers many predictions in the near future, an approach that falls under the generic

class of Long Range Predictive Control (LRPC) algorithms. The high frequency sensitivity is reduced because the control action calculation is overspecified. There are more predicted outputs to be controlled (equations to be solved) than there are permitted control actions (degrees of freedom). The controller therefore must deal with the predicted control errors in an averaging or least-squares way and cannot react too violently to short-term predicted control errors. A thorough overview of the properties of GPC is given in Clarke and Mohtadi (1989).

The GPC cost function at time t, equation 2.3, in its simplest form, is given by:

$$J_{GPC} = \sum_{j=N_1}^{N_2} (y_{sp} - \hat{y}(t+j|t))^2 \tag{4.22}$$

where $\hat{y}(t+j|t)$ is the prediction, or estimate, of $y(t+j)$ based on information available at time t. The prediction, $\hat{y}(t+j|t)$, is therefore the best guess the controller has of the future value of the controlled variable. N_1 and N_2 are called the *prediction horizons*, since the controller does not look beyond N_2 steps into the future and looks no closer than N_1 steps. The cost function is the mathematical expression of the following statement: *The GPC control action at time t is calculated to minimize the second moment of the predicted control error (the sum of squares) over the interval* $t+N_1$ *to* $t+N_2$.

The GPC control action calculation is of course only one part of an adaptive controller. The prediction used by GPC must come from a model, in turn provided by a process identification scheme. Typically some variant of the popular RLS algorithm is used. In the particular case of GPC, process identification was originally added as an afterthought without considering the relevance of RLS to GPC (Mohtadi, 1989). RLS was used almost by default. It is the major point of this work that there exists a more appropriate identification method for adaptive GPC than RLS.

If standard RLS is used with GPC then a number of *ad hoc* modifications must be made to RLS for robust adaptation and control because of the inevitable

structural model-plant mismatch. In other words, because the model *cannot* describe the process completely, RLS must be modified before it can be used safely. One common *ad hoc* fix is the use of a low-pass data pre-filter in process identification in addition to the filtering resulting from the noise model (for example, McIntosh, 1988, Mohtadi, 1990 and E. Lambert, 1987).

Use of such a filter is tantamount to an admission that the noise model is incorrect for the identification even if it is correct for the controller. This is absurd. The identification and control apply to the same process, so the same noise model should be correct for both parts of the adaptive controller.

A new identification method is needed that is based on the same assumptions as the controller. In order to maintain a consistent framework for adaptive controller design, it is best to consider the problem as a whole. An **overall** adaptive control objective is formed and then the identification and control objectives are specified as subproblems. The identification and control then will work hand-in-hand towards a common goal. In this work, the following overall adaptive control objective is used: At time t, the overall control objective is given by:

$$J_{AC} = \sum_{j=N_1}^{N_2} (y_{sp} - y(t+j))^2 \tag{4.23}$$

Obviously $y(t+j)$ is unknown at time t. Nevertheless, this is the goal of the adaptive controller. The adaptive controller must therefore contain the process identification and prediction as part of itself, rather than having them added on in an *ad hoc* way. If the whole cost function is minimized with respect to the model parameter estimates and the control action, then we have a finite horizon dual controller (Feldbaum, 1965).

4.2.1 Dual Control

Feldbaum (1965) describes an adaptive controller structure called *dual control*. Dual control is the optimal solution to the stochastic control problem when the process

information is incomplete. The dual controller is highly nonlinear, and is infeasible to implement, but it defines the best possible adaptive control and hints for a design methodology for more realistic adaptive controllers. A complete discussion of stochastic control theory is beyond the scope of this work; rather the intention is to illustrate some of the consequences of considering an overall control objective.

A dual adaptive controller has to minimize a criterion where the control input, u, has a dual role: to *probe* or excite the process for estimation and yet at the same time exercise *caution* in control due to parameter uncertainty.

The objective of dual control is to minimize the cost function in equation 4.23 subject to a very important assumption: the process parameters are themselves random variables. As with self-tuning controllers, the control problem is divided into estimation and control, but the estimation problem includes finding the conditional probability distribution function of the states and parameters given the process measurements. A Kalman filter can be used for this part of the adaptive controller. In addition, the control action is a nonlinear function of the probability density function — the control action depends on the uncertainties in parameters and states as well as the actual values. A dual controller therefore takes into account parameter uncertainty (cautious control) and measurement uncertainty (stochastic control). The control action eventually applied may also act to reduce parameter uncertainty in order to aid future control calculations (i.e. introduce probing signals). The controller thus exercises both caution and probing.

Dual control is infeasible except in simple examples because of the nonlinear dependence of the control action on the parameters, states and uncertainties (the hyperstate). There are often multiple local minima in the cost function, and the actual global minimum may be difficult to find (Åström and Wittenmark, 1989).

The most interesting property of dual control is its ability to automatically add probing action when the parameters are very uncertain. This is one of the benefits of a long-range control objective: short term losses in control quality (from the

probing action) can be permitted if the control quality in the long run is improved sufficiently. A proper dual controller with a LRPC type objective is mathematically intractable: complex, time-consuming optimization procedures would be necessary to find the optimal control action. Therefore a certainty equivalent GPC controller is considered here. A certainty equivalent controller is one where the controller is designed assuming that the process measurements and model parameters are exactly correct. The need for probing action arises because the parameters are not exactly known, and so probing is contrary to the assumptions underlying certainty equivalence. It is sometimes necessary for the user of GPC to add excitation (through the setpoint or control action and often prior to actual control) but the controller itself cannot do so.[1] The reward of separating the adaptive controller into two parts is that the problem is made tractable.

Adaptive controllers derived from one-step ahead objective functions cannot probe, since the results of the probing would be available too late to affect the one-step ahead control error. Nevertheless, probing has been added to one step ahead adaptive controllers, often in an *ad hoc* way through the addition of a PRBS to the control signal (e.g. Wieslander and Wittenmark, 1971). Many parctical schemes suggest *a priori* probing and estimation before the control loop is closed. There have also been attempts to reduce the *ad hoc* nature of such implementations, through including the covariance matrix in the one step ahead cost function (as discussed in Wittenmark, 1975). In such a method the control action is calculated numerically because there is no closed form solution to the minimization problem. The resulting control calculations are intermediate in complexity between dual control and one-step ahead certainty equivalent control.

Another striking feature of dual control is the objective of the process estimation. The role of the estimation is explicitly to provide the best model for control.

[1]Such user-added excitation may be seen in the experimental data in Chapter 6.

The model quality is judged not on its ability to predict, but on the actual control quality. This is a consequence of the use of an overall control objective. None of the other methods described above (STR, GMV, GPC) considers this. They are straightforward attempts to match the observed process behaviour through matching one-step-ahead predictions. The question of whether or not this is the most relevant objective has been ignored.

4.2.2 Certainty Equivalent Formulation

A certainty equivalent adaptive controller can be formulated using an overall control criterion, equation 4.23. The resulting adaptive controller will be optimal only if the optimal adaptive control problem is certainty equivalent (Patchell and Jacobs, 1971). This seems to be a circular definition, but the term "certainty equivalence" was originally used to describe a class of control *problems*, not a class of controllers. A certainty equivalent control problem is one where (a) the optimal adaptive controller treats the stochastic control problem as a deterministic control problem, i.e. all random variables (both model parameters and signals) may be assumed to be at their mean values during the horizon of interest *without loss of optimality*, and (b) the control problem is *neutral*, i.e. no amount of probing will reduce the model uncertainty. Thus both *caution* and *probing* are absent from a certainty equivalent controller (Jacobs and Patchell, 1972). It should be pointed out that most process control problems are not certainty equivalent, so any certainty equivalent controller will be suboptimal, and therefore adaptive GPC will be a suboptimal controller. Fortunately the goal of this work is not optimal control, but an improvement in process identification for adaptive GPC implementations. Adaptive optimal control is discussed in Bitmead *et al.* (1990).

To introduce some structure to the controller, we recall the definition of $\hat{y}(t+j|t)$, the prediction of $y(t+j)$, given information available at time t, and specify

our controller to be long range predictive in nature.

The error in the prediction, $\epsilon_j(t)$, is then defined as the difference between the actual and predicted values. Although $\epsilon_j(t)$ is not known at time t, the definition will permit the division of the adaptive controller into identification and control tasks.

$$\epsilon_j(t) = y(t+j) - \hat{y}(t+j|t) \tag{4.24}$$

The overall control criterion becomes:

$$J_{AC} = \sum_{j=N_1}^{N_2} (y_{sp} - \hat{y}(t+j|t) - \epsilon_j(t))^2 \tag{4.25}$$

This can be rearranged to give the following three terms:

$$\begin{aligned} J_{AC} &= \sum_{j=N_1}^{N_2} (y_{sp} - \hat{y}(t+j|t))^2 \\ &+ \sum_{j=N_1}^{N_2} (\epsilon_j(t))^2 \\ &-2 \sum_{j=N_1}^{N_2} (y_{sp} - \hat{y}(t+j|t))(\epsilon_j(t)) \end{aligned} \tag{4.26}$$

Certainty Equivalent Controller

The first term on the right hand side of equation 4.26 corresponds exactly to the GPC cost function at time t, as in equation 4.22. GPC is one controller that minimizes the value of this term for a given model, and is one certainty-equivalent control algorithm for the specified adaptive control objective, equation 4.23. The receding horizon formulation for GPC belongs to the class of open loop feedback optimal control methods (see, e.g. Tse and Athans, 1972). A different controller, formulated using closed loop feedback optimal control (dual control) arguments could be used instead, but development of such a controller is beyond the scope of the present work. The reader is directed to Bitmead *et al.* (1990) for a more thorough discussion of adaptive optimal control. Other certainty equivalent receding horizon controllers

exist that minimize this cost function. The differences among such methods are usually results of the process and noise model structure, not design philosophy.

A receding horizon approach permits the use of $NU < N_2$ and improves the controller robustness to severe model-plant mismatch although it does sacrifice closed loop optimality (Clarke *et al.*, 1987a, Ortega and Yang, 1989). The combination of a receding horizon with certainty equivalence produces a controller that from a practitioner's viewpoint is straightforward to implement and yet robust to model-plant mismatch,.

A Control-Relevant Identification Strategy

The second term in equation 4.26 is the identification objective. Expanded, it is given by the following expression.

$$J_{ID} = \sum_{j=N_1}^{N_2} \epsilon_j(t)^2 = \sum_{j=N_1}^{N_2} [y(t+j) - \hat{y}(t+j|t)]^2 \qquad (4.27)$$

In other words, the most relevant identification method for the overall control objective (equation 4.23) will provide the model that predicts best, *not just one step ahead, but also two steps ahead, three, and so on up to N_2 steps ahead.*

This makes sense heuristically, because GPC uses these long range predictions to calculate the control action. The commonly-used Least Squares (LS) method is only concerned with one step ahead predictions and in the presence of structural model plant mismatch, the one-step ahead least squares model will not give the best long range predictions. The use of LS for parameter estimation has limited the effectiveness of adaptive GPC because the models found using LS are not the most appropriate for long range predictive control. Better control could be arrived at through the use of an identification method derived to satisfy the objective in equation 4.27 above.

The implementation and performance of one such method, known as Long

Range Predictive Identification, or LRPI[2], will be discussed in Chapters 5 and 6. An alternative approach to the overall minimization of equation 4.23 will also be discussed in Chapter 7. As for the controller, other parameter estimation methods could be used to minimize the cost in equation 4.27. LRPI is based on LS, but another long range predictive identification method could be formulated using LMS or possibly even a Kalman filter. In addition, the use of multiple models for long range predictive control has been suggested by Mosca and Zappa (1986), but for the present work only GPC, using a single model for all predictions, is considered.

The third term on the right hand side of equation 4.26 is a cross term combining the effects of the identification and control. It is ignored in all certainty equivalent control. The expected value of this term is zero, if the prediction errors for all j are zero mean and uncorrelated with the *predicted* control errors. Because of noise or model-plant mismatch, the prediction errors are normally highly correlated with the actual control errors. They are relatively uncorrelated with the predicted control errors, as long as the model is reasonably accurate. Lu and Fisher (1990) have stated that this term may be considered to be zero on average subject to weak conditions if the model is identified using a regression method, since the prediction errors are orthogonal to the predictions.

4.2.3 An Alternative Formulation

Lu and Fisher (1990) propose an alternative controller structure that also minimizes the overall control objective of equation 4.23. The controller objective is the same as for GPC, but a different predictor is used. A nonminimal predictor is used that provides greater long range predictive accuracy at the expense of more calculation.

The great benefit of the nonminimal predictor structure is that the model that gives the best long range prediction is also the best one step ahead least squares

[2]the name was chosen to emphasize the objective of the identification and illuminate the duality with long range predictive control

model. RLS can therefore be used with no modification to identify the best model for the purposes of control. In effect, the burden of good long range prediction has been passed from the identification to the model. This is certainly an elegant approach. Unfortunately, the number of parameters that must be identified is large, and can be enormous. Nonminimal Predictive Control (NPC), as this approach is called, is discussed at greater length in Chapter 7.

Chapter 5

Long Range Predictive Identification

The case for a control-relevant identification strategy was argued in the last chapter. The focus of the present chapter is to describe the implementation of a feasible GPC-relevant identification scheme. The desirable properties of the identification scheme were that the implementation should be simple, preferably similar to least squares, and that the computational load should not be excessive. In particular, the requirement of further user-specified parameters was to be avoided. Convergence properties are examined through simulation and experimental examples, because of the difficulty of analysis.

5.1 The LRPI Objective

A control-relevant identification approach was introduced in Chapter 4. For GPC the relevant identification objective was shown to be (equation 4.27):

$$J_{ID} = \sum_{j=N_1}^{N_2} ((y(t+j) - \hat{y}(t+j|t))^2$$

The model which minimizes the normal one step ahead least squares cost function (the LS model) will minimize this long range cost only if the process fits within the model structure. If the process is too complex to be represented exactly by the model (i.e. if the model is too simple) then the LS model (which gives the best one step ahead predictions) will not give the best long range predictions. The LS model will therefore not minimize the long range cost. Long Range Predictive Identification (LRPI) is therefore proposed.

The LRPI solution is achieved through a regression approach chosen to be similar to LS. A scalar cost function, J_{LRPI}, is chosen which will in the mean minimize the value of J_{ID}:

$$J_{LRPI} = \frac{1}{t - N_2} \sum_{k=1}^{t-N_2} \left[\frac{1}{N_p} \sum_{j=N_1}^{N_2} \left((y(k+j) - \hat{y}(k+j|k) \right)^2 \right] \tag{5.1}$$

where N_p is the number of predictions considered, i.e. $N_p = N_2 - N_1 + 1$ and $t - N_2$ previous time steps are considered (since $k + j$ must be less than or equal to t). This cost corresponds to the observed mean squared j-step-ahead prediction error. The predictions considered are exactly those which are required by the GPC controller, as was pointed out in Chapter 4. A graphical interpretation of the predictions involved in equation 5.1 is shown in Figure 5.1.

This cost function is attractive because it is a logical extension to long range prediction of the standard least squares regression objective, normally written thus:

$$J_{LS} = \frac{1}{t} \sum_{k=1}^{t} \left[y(k+1) - \hat{y}(k+1|k) \right]^2 \tag{5.2}$$

Although equation 5.1 is conceptually simple, it is computationally difficult to solve for the parameters. The j-step-ahead predictions are nonlinear in the parameters for $j > 1$, so the usual least squares closed form solution is not available. The difficulty is illustrated in the following example.

Example 5.1

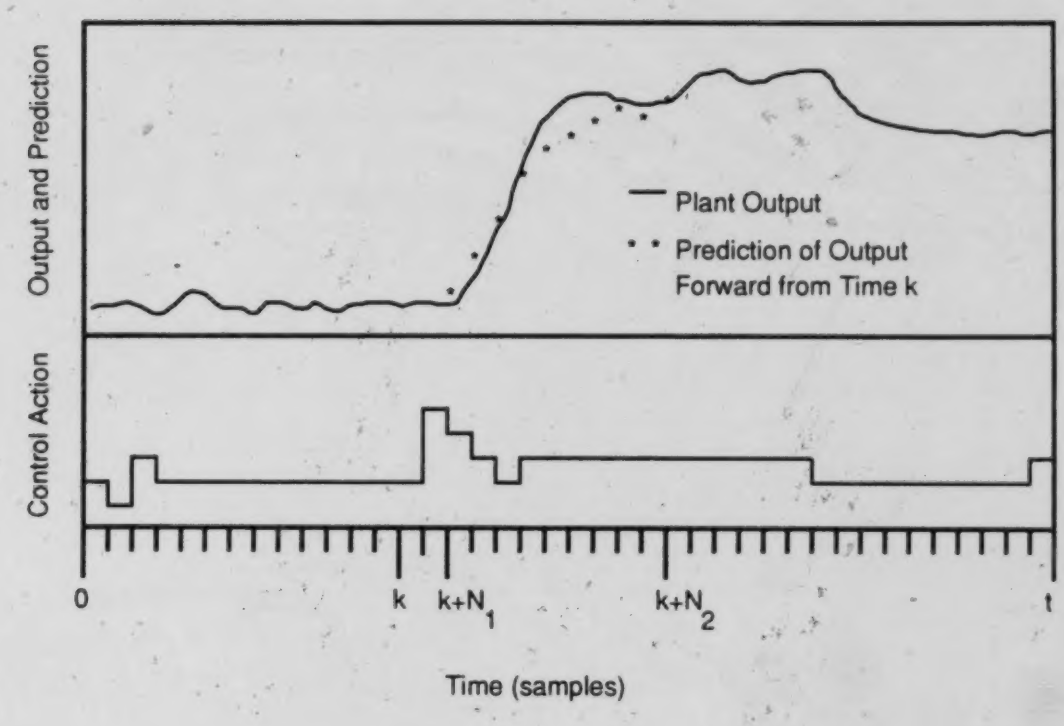

Figure 5.1: Graphical Interpretation of Predictions in LRPI Cost

Assume a deterministic first order process model:

$$y(t) = \hat{a}_1 y(t-1) + \hat{b}_1 u(t-1)$$

The one step ahead prediction is given by:

$$\hat{y}(t+1) = \hat{a}_1 y(t) + \hat{b}_1 u(t)$$

and the two step ahead prediction is given by:

$$\hat{y}(t+2|t) = \hat{a}_1 \hat{y}(t+1|t) + \hat{b}_1 u(t+1)$$

or

$$\hat{y}(t+2|t) = \hat{a}_1^2 y(t) + \hat{a}_1 \hat{b}_1 u(t) + \hat{b}_1 u(t+1)$$

By extension, the j step ahead prediction is

$$\hat{y}(t+j|t) = \hat{a}_1^j y(t) + \hat{a}_1^{j-1} \hat{b}_1 u(t) + \hat{a}_1^{j-2} \hat{b}_1 u(t+1) + \ldots + \hat{b}_1 u(t+j-1)$$

So all predictions are linear in $\hat{b}_1$, but for all $j > 1$ the predictions are nonlinear in $\hat{a}_1$.

If the process model is of higher order then the long range predictions are nonlinear in all parameters including $\hat{b}_i$.

Δ

The method of least squares finds the parameters $\hat{a}_1, \hat{b}_1$ that, for a given set of data, minimize the LS cost function.

If the following definitions are used:

$$Y = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(n) \end{bmatrix}, \Phi = \begin{bmatrix} -y(0) & u(0) \\ -y(1) & u(1) \\ \vdots & \vdots \\ -y(n-1) & u(n-1) \end{bmatrix}, \hat{\theta} = \begin{bmatrix} \hat{a}_1 \\ \hat{b}_1 \end{bmatrix},$$

then LS finds the solution $\hat{\theta}$ to the approximation

$$Y = \Phi\hat{\theta} \tag{5.3}$$

The solution for the batch least squares estimation of $\hat{\theta}$ is:

$$\hat{\theta} = \left[\Phi^T\Phi\right]^{-1}\Phi^T Y \tag{5.4}$$

This solution can only be used if $y(t)$ is linear in the parameters. Otherwise, the problem cannot be written as $Y = \Phi\hat{\theta}$ and this convenient closed form solution cannot be used.

A number of methods are available to minimize the LRPI cost. For batch data, a Newton-Raphson or Gauss-Newton method may be used. Gauss-Newton is more robust, but Newton-Raphson converges faster in the neighborhood of the optimum parameter estimates. For on-line applications, a recursive Gauss-Newton prediction-error method (see e.g. Ljung, 1987) has been used, and has shown reasonable convergence in simulation examples. Although these methods are useful only for batch data, it is instructive to compare the qualities of LRPI and LS models. The cost functions of LS and LRPI also hint at some of the properties of the Newton-Raphson approach.

Example 5.2

The following second order discrete process was simulated.

$$y(t) = -1.4y(t-1) + 0.5y(t-2) + 0.1u(t-1) + 0.05u(t-2)$$

The process input, u, was white noise. A first order model structure

$$y(t) = \hat{a}_1 y(t-1) + \hat{b}_1 u(t-1)$$

was chosen. Because of the high degree of excitation and the structural mismatch between the process (second order) and the model (first order),

there is no set of model parameters that will provide perfect prediction, even though the process is deterministic.

The LRPI cost for $N_1 = 1$, $N_2 = 10$ was calculated for different parameter choices over the range $\hat{a}_1 \in [-2, +1], \hat{b}_1 \in [-1, +1]$. Figure 5.2 shows how the value of J_{LRPI} changes as a function of the model parameters. The contours are logarithmically spaced because of the extremely high cost in the area $\hat{a}_1 < -1$.

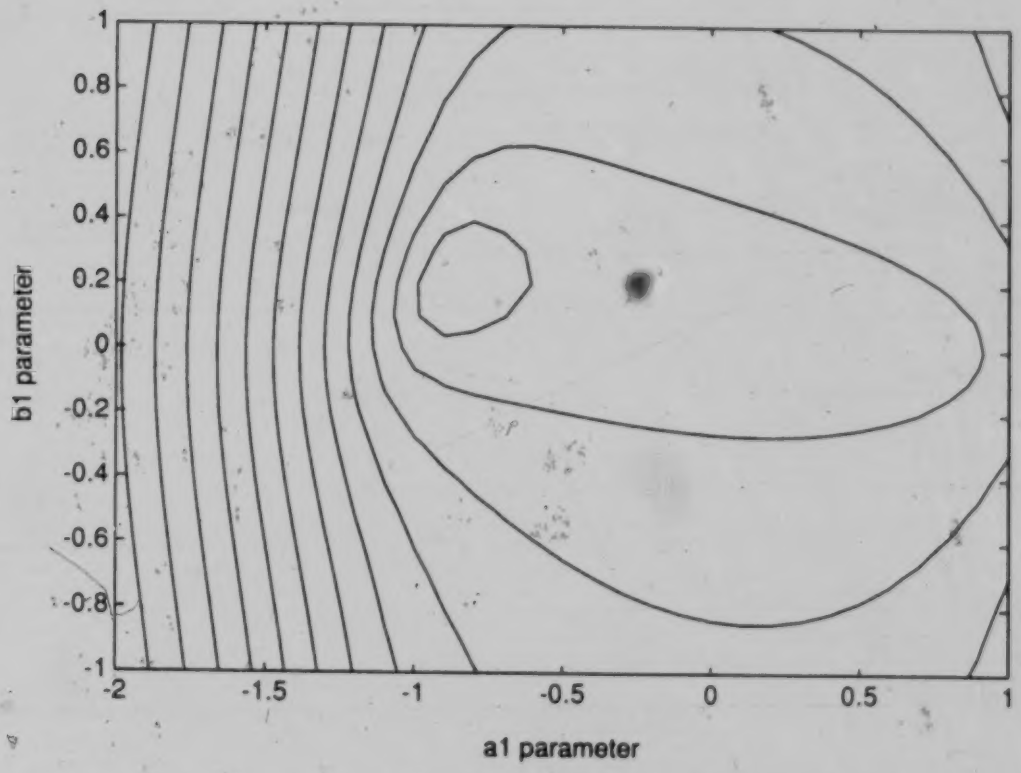

Figure 5.2: Contour Plot of LRPI Cost as a Function of Parameter Estimates

The cost is unimodal, and is quite a smooth function of the parameter estimates within the region $-1.0 < \hat{a}_1 < 0.0$, which corresponds to a stable process. For values of $\hat{a}_1$ less than -1, the model is unstable and the cost increases dramatically because of the poor long range predictions. This signifies a large implicit penalty in LRPI for $\hat{a}_1 < -1$. A similar

effect could also be obtained in LS by imposing an external constraint on the optimization, using, for example, Lagrange multipliers. The optimum would then remain at the LS optimum unless it violated the constraint. The LS and LRPI optima are at different locations, as can be seen in Table 5.1, so a hard constraint would only serve to prevent totally unreasonable parameters. When $\hat{b}_1$ becomes negative (negative process gain), the cost also increases quickly, although not as dramatically as for an unstable model. The long range predictions depend more on the quality of the autoregressive parameter estimate than on the quality of the estimate of the gain.

By contrast, the shape of the loss function for a least squares (one step ahead prediction) model is a simple paraboloid. The cost does not increase as quickly as for LRPI in the regions corresponding to negative gain or unstable models, as can be seen in Figure 5.3. Once again the contours are logarithmically spaced, but the entire range of cost is much smaller for LS than for LRPI, as can be seen in Table 5.1. The three contours closest to the optimum in Figure 5.2 cover the entire range of values in Figure 5.3.

Table 5.1: Summary of Cost Function Information for Example 5.2

Method	LS	LRPI
minimum cost	0.0115	0.0339
maximum cost	1.9522	7.4×10^4
optimum $\hat{a}_1$	-0.9475	-0.8815
optimum $\hat{b}_1$	0.1001	0.2021
cost at LS optimum	0.0115	0.0714
cost at LRPI optimum	0.0217	0.0339

The LS cost is smaller than the LRPI cost at the LRPI optimum (in

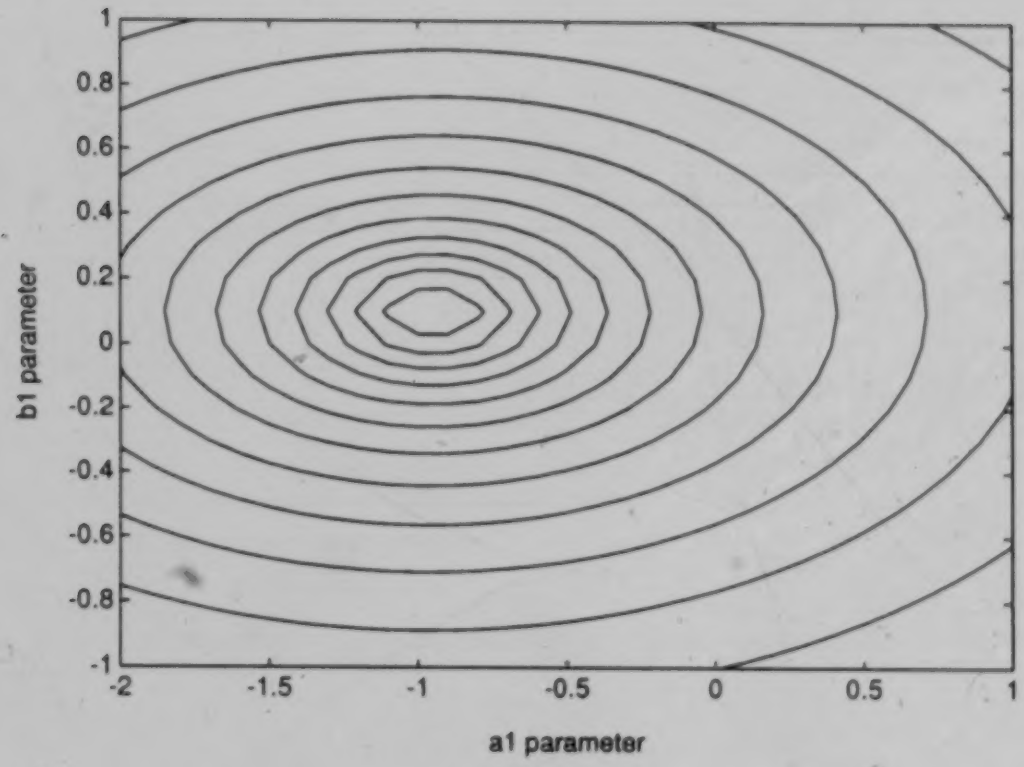

Figure 5.3: Contour Plot of LS Cost as a Function of Parameter Estimates

Table 5.1) because LS is only concerned with matching one prediction while LRPI must approximately match several. The mean squared one step ahead prediction error at the LRPI optimum (0.0217) is less than the mean square of the $1, 2, 3, \ldots, 10$ step ahead prediction errors (0.0339).

The step responses of the LS and LRPI optimum models are shown in Figure 5.4. It is clear that the LRPI model provides a better match to the step response over the first 10 steps than the LS model. In particular, the LS model fits the first point well, but later ones poorly. The step responses are shown for two reasons: first, they are simple, easily understood examples of long range prediction, and secondly, the step response parameters are themselves important to GPC.

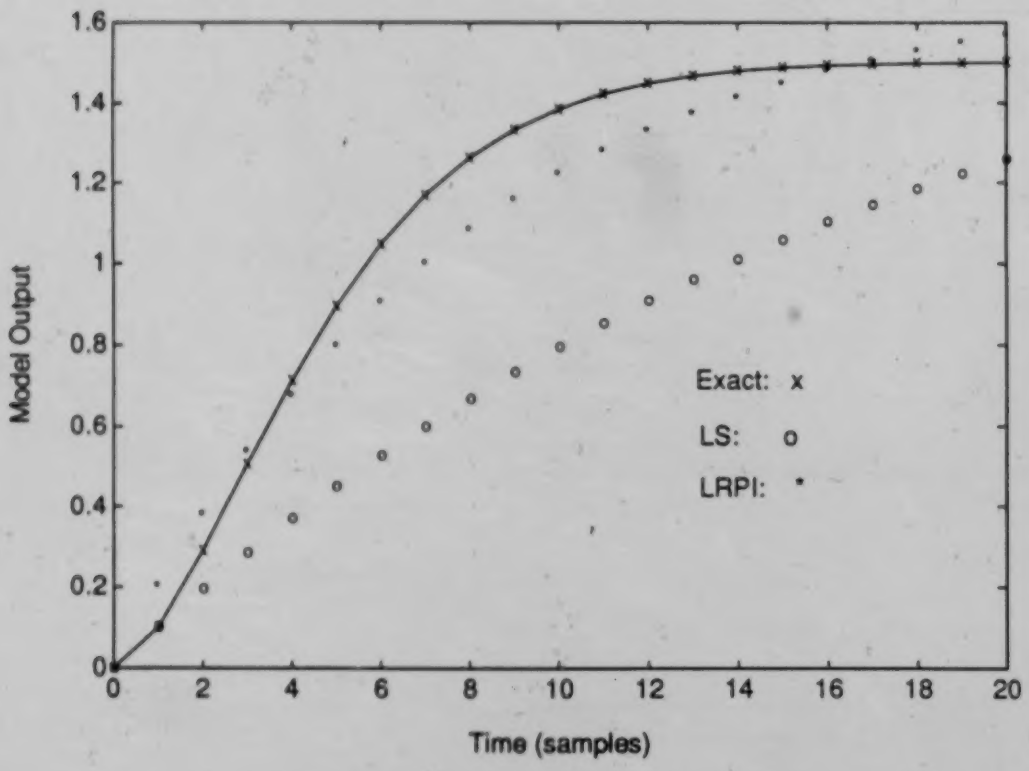

Figure 5.4: Step Responses of LS and LRPI Models

When the GPC control action is calculated, a vector of future errors

is multiplied by the pseudoinverse of a matrix of step response coefficients from N_1 to N_2. If these values are badly underestimated (as for the LS model), the resulting controller will have too high a gain, giving oscillations and instability. The more accurate estimates from the LRPI model will result in better control, for this example.

Δ

The exact solution of equation 5.1 in the batch and recursive cases is described in section 5.2. The frequency domain implications of LRPI and the resulting recursive implementation are discussed in section 5.3.

5.2 Exact Solution of the LRPI Problem

As was stated above, there is no closed form solution to the minimization of the LRPI cost function, and therefore an iterative technique must be used. To find the parameter values which minimize the cost function, equation 5.1, the j-step-ahead predictions must first be defined in terms of the process model parameters.

The GPC predictor is:

$$\hat{y}(t+j|t) = F_j(q^{-1})y(t) + E_j(q^{-1})B(q^{-1})\Delta u(t+j-1) \tag{5.5}$$

where $F_j(q^{-1})$ and $E_j(q^{-1})$ are given by the Diophantine equation:

$$T(q^{-1}) = E_j(q^{-1})\hat{A}(q^{-1})\Delta + q^{-j}F_j(q^{-1}) \tag{5.6}$$

(The denominator of the noise model, Δ, a specific case and describes the nonstationarity of the disturbances. More generally, Δ can be replaced by a polynomial $D(q^{-1})$ which may contain Δ as a factor. For example, $D(q^{-1})$ may be equal to Δ^2 in some robotic applications. See section 5.3.4)

We rewrite equation 5.1 as a minimization problem:

$$\hat{\theta} = \arg\ \min \frac{1}{t-N_2} \sum_{k=1}^{t-N_2} \left[\frac{1}{N_p} \sum_{j=N_1}^{N_2} ((y(k+j) - \hat{y}(k+j|k))^2 \right] \tag{5.7}$$

where $\hat{\theta}$ is the vector of parameter estimates:

$$\hat{\theta} = [\hat{a}_1, \ldots, \hat{a}_{n_a}, \hat{b}_1, \ldots, \hat{b}_{n_b}]^T \tag{5.8}$$

5.2.1 Batch Solution

Newton-Raphson

One simple way to solve this minimization is to use Newton-Raphson to find the location (parameter vector) where all first derivatives of J_{LRPI} with respect to the parameters are zero. (All such points are either minima, maxima or saddle points. The problem of avoiding maxima and saddle points will be ignored because the minimization is well-behaved in the neighbourhood of the optimum.) The minimization problem is thus changed to a multivariate root-finding problem. Newton-Raphson is intuitively appealing because of its relatively simple formulation and excellent convergence properties.

The derivative vector is given by

$$\frac{\partial J_{LRPI}}{\partial \hat{\theta}} = \frac{1}{t-N_2} \sum_{k=1}^{t-N_2} \left[\frac{1}{N_p} \sum_{j=N_1}^{N_2} ((y(k+j) - \hat{y}(k+j|k)) \frac{\partial \hat{y}(k+j|k)}{\partial \hat{\theta}} \right] \tag{5.9}$$

and the Hessian (the matrix of second derivatives) is:

$$\begin{aligned} \mathbf{H} = \frac{\partial}{\partial \hat{\theta}} \left(\frac{\partial J_{LRPI}}{\partial \hat{\theta}} \right)^T = \\ \frac{1}{t-N_2} \sum_{k=1}^{t-N_2} \frac{1}{N_p} \sum_{j=N_1}^{N_2} \left[((y(k+j) - \hat{y}(k+j|k)) \frac{\partial^2 \hat{y}(k+j|k)}{\partial \hat{\theta}^2} \right. \\ \left. + \left(\frac{\partial \hat{y}(k+j|k)}{\partial \hat{\theta}} \right)^T \left(\frac{\partial \hat{y}(k+j|k)}{\partial \hat{\theta}} \right) \right] \end{aligned} \tag{5.10}$$

The standard Newton update is used.

$$\hat{\theta} = \hat{\theta} - (\mathbf{H})^{-1}\left(\frac{\partial J_{LRPI}}{\partial \hat{\theta}}\right) \tag{5.11}$$

The actual difficulty is twofold. First, the number of derivatives to be calculated is enormous: $N_p \times (t - N_2) \times (na + nb)$ first *and* second derivatives are required. Secondly, it is not easy to find a general form for calculation of the derivatives. LS is much simpler: all first derivatives are simply measurements (and therefore need no calculation) and all second derivatives are zero.

Example 5.3

As an illustration, consider the j-step ahead predictor of Example 5.1.

The j step ahead predictor is

$$\hat{y}(t+j|t) = \hat{a}_1^j y(t) + \hat{a}_1^{j-1}\hat{b}_1 u(t) + \hat{a}_1^{j-2}\hat{b}_1 u(t+1) + \ldots + \hat{b}_1 u(t+j-1)$$

The first derivatives are:

$$\begin{aligned}
\frac{\partial \hat{y}(t+j|t)}{\partial \hat{a}_1} &= j\hat{a}_1^{j-1}y(t) + (j-1)\hat{a}_1^{j-2}\hat{b}_1 u(t) \\
&\quad +(j-2)\hat{a}_1^{j-3}\hat{b}_1 u(t+1) + \ldots + 1\hat{a}^0\hat{b}_1 u(t+j-2) \\
\frac{\partial \hat{y}(t+j|t)}{\partial \hat{b}_1} &= \hat{a}_1^{j-1}u(t) + \hat{a}_1^{j-2}u(t+1) + \ldots + u(t+j-1)
\end{aligned}$$

and the second derivatives are:

$$\begin{aligned}
\frac{\partial^2 \hat{y}(t+j|t)}{\partial \hat{a}_1^2} &= (j)(j-1)\hat{a}_1^{j-2}y(t) + (j-1)(j-2)\hat{a}_1^{j-3}\hat{b}_1 u(t) + \ldots \\
&\quad +(2)(1)\hat{a}^0\hat{b}_1 u(t+j-3) \\
\frac{\partial^2 \hat{y}(t+j|t)}{\partial \hat{b}_1^2} &= 0 \\
\frac{\partial^2 \hat{y}(t+j|t)}{\partial \hat{a}_1 \partial \hat{b}_1} &= (j-1)\hat{a}_1^{j-2}u(t) + (j-2)\hat{a}_1^{j-3}u(t+1) + \ldots \\
&\quad +u(t+j-2)
\end{aligned}$$

Note that this is the simplest form of long range prediction: there is no noise model or filtering to complicate the issue.

Derivatives of higher order models are much more complicated. For second order models there are four first order derivatives and sixteen (ten unique) second derivatives. The derivatives must be worked out for every model structure individually; there is no useful general form.

△

The shape of the cost function surface affects the convergence properties of any method used to find the optimal parameters. In particular Newton-Raphson converges in one iteration when the cost function is a paraboloid, but it takes significantly longer when the shape is as in Figure 5.2. Typically for small N_2 (less than 10) fewer than N_2 iterations are required. In most cases, speed of convergence could be improved through the use of a modified Newton scheme, such as the well-known Marquardt-Levenberg method.

Convergence can be a problem with noisy data or when there is significant structural model-plant mismatch. In such cases, the least squares parameters may "wander" — the noise, input sequence and high frequency dynamics may cause changes in the position of the minimum. It is often possible for the optimal LS $\hat{a}_1$ parameter to have a value very close to -1.0 for a first order model. It may then wander temporarily beyond that value, i.e. into the unstable region. The resulting model usually results in poor or unstable control, in part because the model is unstable, and also because the steady state gain has the wrong sign. (The model steady state gain is given by the ratio $\frac{\hat{B}(1)}{\hat{A}(1)}$, and if $\hat{a}_1$ is less than -1 for a first order model, then $\hat{A}(1) < 0$.) The optimal LRPI model parameters do not wander as far as the LS parameters because of the drastic increase in the cost beyond $\hat{a}_1 = -1$.

The quality of long range predictions is profoundly affected by $\hat{a}_1$. The accuracy of the autoregressive part of the model is tried when long-range forecasting is required. LRPI therefore reduces the likelihood of accidentally identifying an unstable model when the process is stable. Least squares cannot take into account the

need for good long range predicitons.

Gauss-Newton

The robustness of LRPI may be improved and its complexity reduced at the expense of speed of convergence by using a Gauss-Newton algorithm rather than a Newton-Raphson. Gauss-Newton is similar, but the second derivative term is removed from the Hessian. The Hessian, **H** is then approximated as:

$$\mathbf{H} \approx \frac{1}{t-N_2} \sum_{k=1}^{t-N_2} \left[\frac{1}{N_p} \sum_{j=N_1}^{N_2} \left(\frac{\partial \hat{y}(k+j|k)}{\partial \hat{\theta}} \right)^T \left(\frac{\partial \hat{y}(k+j|k)}{\partial \hat{\theta}} \right) \right] \tag{5.12}$$

Use of Gauss-Newton reduces the number of calculations required per iteration as well as simplifying the formulation: no second derivatives need to be calculated. One additional advantage of Gauss-Newton is that the Hessian is almost guaranteed to be positive definite for all parameter values. As long as the number of observations is greater than N_P, and the data are persistently exciting, then the Hessian will be positive definite.

Convergence is slower than for Newton-Raphson, being between linear and quadratic, while Newton-Raphson typically shows quadratic convergence. For batch calculations this is not much of an issue: the increase in the number of iterations is offset by the reduction in the number of calculations per iteration, and convergence tends to speed up near the optimum as the neglected terms in the Hessian diminish.

Gauss-Newton is also far more useful in recursive implementations than Newton-Raphson, since it can be quite easily formulated in a prediction error form.

5.2.2 Recursive Solution

For on-line applications, a recursive Gauss-Newton prediction error method may be used (Ljung and Sōderstrōm, 1983). It is similar to the standard RLS method, but the approximated Hessian replaces the covariance matrix. The standard RLS update

for one time step is given by the following:

$$\mathbf{P}(t) = \mathbf{P}(t-1) - \frac{\mathbf{P}(t-1)\phi(t)\phi^T(t)\mathbf{P}(t-1)}{1+\phi^T(t)\mathbf{P}(t-1)\phi(t)} \tag{5.13}$$

$$e(t) = y(t) - \hat{y}(t) \tag{5.14}$$

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \mathbf{P}(t)\phi(t)e(t) \tag{5.15}$$

where $\mathbf{P}(t)$ is the covariance matrix, defined as:

$$\mathbf{P}^{-1}(t) = \sum_{k=0}^{t} \phi(k)\phi^T(k) \tag{5.16}$$

and $\phi(t)$ is the *regressor* or data vector, which in example 5.2 is:

$$\phi(t) = \begin{bmatrix} -y(t-1) \\ u(t-1) \end{bmatrix} \tag{5.17}$$

For recursive Gauss-Newton, the covariance matrix is replaced by the inverse of the Hessian, and ϕ, the regressor vector, is replaced with the vector of first derivatives of $\hat{y}$ with respect to $\hat{\theta}$. The Hessian update at time t is given by the following relation:

$$\mathbf{H}(t) = \mathbf{H}(t-1) + \frac{1}{N_p}\sum_{j=N_1}^{N_2}\left(\frac{\partial\hat{y}(t-N_2+j|t-N_2)}{\partial\hat{\theta}}\right)^T\left(\frac{\partial\hat{y}(t-N_2+j|t-N_2)}{\partial\hat{\theta}}\right) \tag{5.18}$$

Updating the inverse of the Hessian for each time step calls for the equivalent of $N_p = (N_2 - N_1 + 1)$ different covariance matrix updates. Moreover, there are N_p different prediction errors and different derivative vectors. Recursive Gauss-Newton therefore calls for at least N_p times as many calculations as RLS. It is possible to reduce the number of calculations required somewhat through approximations or a single higher-rank matrix update but the computational load remains heavy.

On-line convergence is slowed also because the approximation of the Hessian is even coarser than before. Since the derivatives are functions of $\hat{\theta}$, $\mathbf{H}$ should change if the parameter estimates change, even if no new measurements are taken. The

computational load for doing so would be equal to that for performing a complete batch Gauss-Newton calculation each time step, which is infeasible. The Hessian update only takes the present parameter values into account for the new data: the existing Hessian is not corrected first. Consequently convergence is slowed (although a forgetting factor may help matters by discarding old, outdated values).

5.3 Implementation Through Adaptive Filtering

The recursive form of LRPI has *prima facie* a very poor case. It is computationally demanding and has no known useful convergence properties, even in the ideal noise free environment. The goal of control-relevant identification however is sufficiently strong to point us to an alternative analysis.

Wahlberg and Ljung (1986) and Ljung (1987) analyze the asymptotic properties of the least squares estimate in the frequency domain. Their methods of analysis involve the re-casting of the identification objective from the time domain to the frequency domain. The characteristics of signals and processes may then be examined in the frequency domain, rather than in the time or z domain.

The topic of frequency analysis is not new to control, nor indeed to engineering mathematics in general. Fourier series are often used in the solution of differential equations, and both continuous and discrete transfer functions are commonly represented by Bode or Nyquist diagrams. The topic of spectral analysis of signals is rarely encountered in chemical engineering, and yet spectral analysis can yield surprising results, as will be seen later in this section.

5.3.1 Frequency Domain Analysis of Cost Function

To simplify the discussion, analysis here will be limited to the deterministic case with structural model-plant mismatch usually caused by underestimating the order of the process transfer function. Extensions to the stochastic environment are discussed in

Section 5.3.4. The process is assumed to be of the form

$$y(t) = \frac{B^0(q^{-1})}{A^0(q^{-1})}u(t-1) \tag{5.19}$$

where $A^0(q^{-1})$ and $B^0(q^{-1})$ are polynomials in the backshift operator, q^{-1}. The process model is given by Clarke *et al.* (1987b) as:

$$\hat{A}(q^{-1})y(t) = \hat{B}(q^{-1})u(t-1) + \frac{T(q^{-1})}{\Delta}\xi(t)$$

where $\hat{A}$ and $\hat{B}$ are in this case lower order polynomials than A^0 and B^0 respectively. The noise model $(T(q^{-1})/\Delta)\xi(t)$ is imposed by GPC.

The j-step-ahead GPC predictor is (equation 5.5):

$$\hat{y}(t+j|t) = \frac{F_j}{T}y(t) + \frac{E_j\hat{B}}{T}\Delta u(t+j-1)$$

The LRPI cost function, J_{LRPI}, may be rewritten in terms of the GPC predictor:

$$J_{LRPI} = \frac{1}{t-N_2}\sum_{k=1}^{t-N_2}\frac{1}{N_p}\sum_{j=N_1}^{N_2}\left[y(k+j) - \left(\frac{F_j y(k)}{T} + \frac{E_j\hat{B}}{T}\Delta u(k+j-1)\right)\right]^2 \tag{5.20}$$

The j step ahead prediction error at time k (using filtered data) is contained within the large square brackets. To analyze the dependence of the model $\{\hat{A}, \hat{B}\}$ on the design variables, it is first desirable to remove the process output, y, from equation 5.20.

The Diophantine equation (5.6) may be rearranged thus:

$$1 - q^{-j}\frac{F_j(q^{-1})}{T(q^{-1})} = \frac{E_j(q^{-1})\hat{A}(q^{-1})\Delta}{T(q^{-1})}$$

and this may be applied to $y(k+j)$ in the cost function to give:

$$J_{LRPI} = \frac{1}{t-N_2}\sum_{k=1}^{t-N_2}\frac{1}{N_p}\sum_{j=N_1}^{N_2}\left[\frac{E_j\hat{A}\Delta}{T}y(k+j) - \frac{E_j\hat{B}}{T}\Delta u(k+j-1)\right]^2 \tag{5.21}$$

The true process description, equation 5.19, may now be used to express $y(k+j)$ in terms of $u(k+j-1)$:

$$J_{LRPI} = \frac{1}{t-N_2}\sum_{k=1}^{t-N_2}\frac{1}{N_p}\sum_{j=N_1}^{N_2}\left[\frac{E_j\hat{A}\Delta}{T}\frac{B^0}{A^0}u(k+j-1) - \frac{E_j\hat{B}}{T}\Delta u(k+j-1)\right]^2 \tag{5.22}$$

This can finally be rearranged to give the following equation.

$$J_{LRPI} = \frac{1}{t-N_2}\sum_{k=1}^{t-N_2}\frac{1}{N_p}\sum_{j=N_1}^{N_2}\left[\left(\frac{E_j\hat{A}\Delta}{T}\right)\left(\frac{B^0}{A^0} - \frac{\hat{B}}{\hat{A}}\right)u(k+j-1)\right]^2 \tag{5.23}$$

The contents of the large square brackets are still the j step ahead filtered prediction error at time k, as in equation 5.20.

We now have an expression that gives the mean square prediction error as an explicit function of the design variables N_1 and N_2, the noise model $T(q^{-1})$, the actual process, the process model and the input sequence. The LS cost is the same, but with N_1 and N_2 both equal to 1:

$$J_{LS} = \frac{1}{t}\sum_{k=1}^{t}\left[\left(\frac{\hat{A}\Delta}{T}\right)\left(\frac{B^0}{A^0} - \frac{\hat{B}}{\hat{A}}\right)u(k-1)\right]^2 \tag{5.24}$$

The noise model is present in both of these expressions as the Δ/T term. If, in addition to the noise model, a filter, $L(q^{-1})$, were used for identification only, then the LS cost function would change to the following:

$$J_{LS} = \frac{1}{t}\sum_{k=1}^{t}\left[\left(\frac{L\hat{A}\Delta}{T}\right)\left(\frac{B^0}{A^0} - \frac{\hat{B}}{\hat{A}}\right)u(k-1)\right]^2 \tag{5.25}$$

To compare equations 5.23 and 5.25 in a meaningful way, it is useful to examine the frequency domain properties of the different models. The frequency domain provides a useful framework for analyzing different processes and models regardless of their orders. It is difficult to compare, for instance, a second order process and a first order model through examining the parameters themselves. Typically step or impulse responses are used for comparison. They are however incomplete. A step

response contains relatively little information about the response at high frequencies, and an impulse response does not show the steady state gain well. A single Bode plot completely describes any transfer function.

The discussion so far has focussed on prediction errors. The models are formed by minimizing certain functions of prediction errors. In particular, squared prediction errors. If the one step ahead prediction error, $\epsilon_1(t)$ is thought of as a signal or time series, then the LS cost function, equation 5.2 is just the mean square value of this signal, or the mean power in the signal. The power or energy in a signal can be measured directly:

$$P(\epsilon) = \frac{1}{t}\sum_{k=1}^{t}\epsilon(k)^2$$

or it can be measured from the *power spectrum:*

$$P(\epsilon) = \frac{h}{2\pi}\int_{-\pi/h}^{\pi/h}\Phi_\epsilon(\omega)d\omega$$

where h is the sample time and $\Phi_\epsilon(\omega)$ is the power spectrum of ϵ. The value of Φ_ϵ at a given frequency ω_0 represents the power of the signal at that frequency. If the signal ϵ were filtered perfectly with an ideal band-pass filter passing through only at frequency ω_0, then the power in the filtered signal would be equal to $\Phi_\epsilon(\omega_0)$. It is more convenient to use the power spectrum here than the more straightforward Fourier transform because these cost functions are written in terms of squared errors.

A full introduction to Fourier transforms and power spectra can be found in any signal processing or spectral analysis textbook, e.g. Priestley (1981). The only important lemma required for discussion here is the following, stated here without proof.

Lemma 1 *Given a (quasistationary) signal $e(t)$ with power spectrum $\Phi_e(\omega)$, a stable transfer function $G(q^{-1})$, and $y(t)$ defined as follows:*

$$y(t) = G(q^{-1})e(t)$$

then $\Phi_y(\omega)$ is given by:

$$\Phi_y(\omega) = |G(e^{i\omega})|^2 \Phi_e(\omega)$$

The expression $|G(e^{i\omega})|$ is just the amplitude ratio (also called magnitude) of $G(q^{-1})$.

The proof for this can be found in Appendix 2A of Ljung (1987). It permits us to examine the frequency domain transformations of equations 5.23 and 5.25. Making use of this, it can fairly easily be shown that the LS cost function, equation 5.25, may be written as follows:

$$J_{LS} = \frac{h}{2\pi} \int_{-\pi/h}^{\pi/h} \left| \frac{L\hat{A}\Delta}{T} \right|^2 \left| \frac{B^0}{A^0} - \frac{\hat{B}}{\hat{A}} \right|^2 \Phi_u(\omega) d\omega \qquad (5.26)$$

where $\Phi_u(\omega)$ is the power spectrum of the input, $u(t)$.

The LRPI cost function may in turn be written as the sum of N_p different j step ahead cost functions:

$$J_{LRPI} = \frac{1}{N_p} \sum_{j=N_1}^{N_2} \left[\frac{h}{2\pi} \int_{-\pi/h}^{\pi/h} \left| \frac{E_j \hat{A}\Delta}{T} \right|^2 \left| \frac{B^0}{A^0} - \frac{\hat{B}}{\hat{A}} \right|^2 \Phi_u(\omega) d\omega \right] \qquad (5.27)$$

The summation may be moved within the integral:

$$J_{LRPI} = \frac{h}{2\pi} \int_{-\pi/h}^{\pi/h} \frac{1}{N_p} \sum_{j=N_1}^{N_2} \left| \frac{E_j \hat{A}\Delta}{T} \right|^2 \left| \frac{B^0}{A^0} - \frac{\hat{B}}{\hat{A}} \right|^2 \Phi_u(\omega) d\omega \qquad (5.28)$$

What is interesting about this equation is that the only term that explicitly depends on the prediction horizons (must be within the summation) is the magnitude of E_j. To make this clear, equation 5.28 may be rewritten as:

$$J_{LRPI} = \frac{h}{2\pi} \int_{-\pi/h}^{\pi/h} \left| \frac{\hat{A}\Delta}{T} \right|^2 \left| \frac{B^0}{A^0} - \frac{\hat{B}}{\hat{A}} \right|^2 \Phi_u(\omega) \frac{1}{N_p} \sum_{j=N_1}^{N_2} |E_j|^2 d\omega \qquad (5.29)$$

The only difference between the LS and LRPI costs is the L filter in the LS equation and the summation of E_j in the LRPI cost. If all other parts of the cost

functions are the same (same input, same model, same actual process) then the LS cost is equal to the LRPI cost if the following condition holds:

$$|L(\omega)|^2 = \frac{1}{N_p} \sum_{j=N_1}^{N_2} |E_j(\omega)|^2 \qquad (5.30)$$

for all ω.

Conversely, if the LS method is implemented with an $L(q^{-1})$ filter that meets this requirement, then the resulting model will be the same as the LRPI model.

So, LRPI may be implemented using a form of LS, but with an additional filter, $L(q^{-1})$, the form of which is determined by N_1, N_2 and E_j. E_j in turn is given by the Diophantine equation (5.6), and depends on the process and noise model. The problem now is to find the L filter.

Fortunately, there is a unique $L(q^{-1})$ of order N_2 which satisfies the criterion (5.30). The way to find it is by the method of spectral factorization, for instance the technique of Bohm *et al.* (1984). For on-line applications, only one iteration of the method is carried out each time step, as convergence is quite rapid. The actual use of spectral factorization for LRPI is described in Section 5.3.2. The properties of the $L(q^{-1})$ filter are then discussed in Section 5.3.3.

5.3.2 Spectral Factorization

The LRPI filter, $L(q^{-1})$, is defined in the frequency domain by equation 5.30:

$$|L(\omega)|^2 = \frac{1}{N_p} \sum_{j=N_1}^{N_2} |E_j(\omega)|^2$$

for all ω.

The transformation of any polynomial in q or q^{-1} into the frequency domain is achieved through the use of the following identity:

$$q = e^{j\omega}$$

so $L(\omega)$ is more correctly written as $L(e^{-j\omega})$.

The characteristics of $L(e^{-j\omega})$ as a function of ω are given below:

$$\left|L(e^{-j\omega})\right| = \sqrt{\mathrm{Re}\left(L(e^{-j\omega})\right)^2 + \mathrm{Im}\left(L(e^{-j\omega})\right)^2}$$

$$\phi(L(e^{-j\omega})) = \tan^{-1}\frac{\mathrm{Im}\left(L(e^{-j\omega})\right)}{\mathrm{Re}\left(L(e^{-j\omega})\right)}$$

In other words, the magnitude of $L(e^{-j\omega})$ at a given frequency ω_0 is the absolute value of $L(e^{-j\omega_0})$ and the phase angle of L is the inverse tangent of the imaginary part divided by the real part.

The square of the magnitude can be calculated by multiplying $L(e^{-j\omega})$ by its complex conjugate: $L(e^{+j\omega})$. In the polynomial domain $L(e^{+j\omega})$ is $L(q^{+1})$.

So, equation 5.30 becomes:

$$L(e^{-j\omega})L(e^{j\omega}) = \frac{1}{N_p}\sum_{j=N_1}^{N_2} E_j(e^{-j\omega})E_j(e^{+j\omega}) \tag{5.31}$$

$$L(q^{-1})L(q^{+1}) = \frac{1}{N_p}\sum_{j=N_1}^{N_2} E_j(q^{-1})E_j(q^{+1}) \tag{5.32}$$

To find $L(q^{-1})$ the right hand side of equation 5.32 is calculated. The result is called the spectrum. The spectrum must then be factored to find the stable root, $L(q^{-1})$. The method used for LRPI is the simple method of Bohm *et al.* (1984). It works as follows.

Given a spectrum $S(q)$ (in this case equal to $L(q)L(q^{-1})$) and an approximate (stable) factor $F_n(q)$, perform the polynomial division (deconvolution):

$$\frac{S(q)}{F_n(q)} = F_{n+1}(q^{-1}) + \frac{R(q^{-1})}{F_n(q)}$$

$F_{n+1}(q^{-1})$ is the new approximate factor and $R(q)$ is the remainder. The notation here is important. $F(q)$ and $F(q^{-1})$ are not the same polynomial. One is written in ascending powers of q and the other is in ascending powers of q^{-1}. For example, if

$$F_n(q^{-1}) = 1 + f_{n1}q^{-1} + f_{n2}q^{-2}$$

then

$$F_n(q) = 1 + f_{n1}q^{+1} + f_{n2}q^{+2}$$

which is something altogether different.

To perform another iteration it is necessary to form $F_{n+1}(q)$ from $F_{n+1}(q^{-1})$. As many iterations are carried out as wished; the magnitude of $R(q)$ may be used as a convergence criterion.

Δ

For the first step, $S(q)$ is just the spectrum, the right hand side of equation 5.32, and $F_0(q)$ may be chosen to be 1.0 or any other stable polynomial. Convergence in practice is rapid and the method is robust. There are other spectral factorization methods, but any improvement in performance was not felt to be worth the extra programming effort.

In experimental practice, only one iteration is performed each time step. $S(q)$ is calculated anew from the updated parameters and the old L filter is used as $F_n(q^{-1})$. There has been no trouble with convergence of the L filter using this method.

5.3.3 Properties of the Adaptive Filter

The following properties of the LRPI L filter have been observed.

Observation 1 : *$L(q^{-1})$ is a function of the controller tuning and the process model.*

From the definition of $L(q^{-1})$, equation 5.32:

$$L(q^{-1})L(q^{+1}) = \frac{1}{N_p}\sum_{j=N_1}^{N_2} E_j(q^{-1})E_j(q^{+1})$$

it is clear that L depends on the controller prediction horizons N_1 and N_2. And, $E_j(q^{-1})$ is a function of both the process model and the noise model, so it follows that $L(q^{-1})$ is too.

Observation 2 : *For all open loop stable process models i.e.*

$$\hat{A}(q^{-1}) \neq 0, \qquad \forall |q| \geq 1$$

and $N_2 = 1$, $L = 1$ we have an equation error least squares method. When the disturbance model denominator $D(q^{-1})$ is stable, or for positional identification (where $D(q^{-1}) = 1$), then as N_2 tends to infinity the estimator tends from an equation error scheme toward an output error one.

The first part of this is clear: when $N_2 = 1$ LRPI reduces to LS, as has been pointed out above. However, the implications of the second part are somewhat more complicated.

Recalling the Diophantine equation defining $E_j(q^{-1})$,

$$T(q^{-1}) = E_j(q^{-1})\hat{A}(q^{-1})D(q^{-1}) + q^{-j}F_j(q^{-1})$$

Solving for E_j:

$$E_j = \frac{T}{\hat{A}D} - q^{-j}\frac{F_j}{\hat{A}D}$$

As N_2 tends to infinity, $E_{N_2}(q^{-1})$ becomes infinite in length and becomes a closer and closer approximation to $T/\hat{A}D$. In the limit, $L(q^{-1})$ also becomes equal to $T/\hat{A}D$. This is because L is a sort of average of all E_j and in the limit there is an infinite number of E_j polynomials equal to $T/\hat{A}D$.

The measurements are filtered by LD/T (as ever) and so the net effect of the combined filtering is

$$\frac{LD}{T}u(t) = \frac{(T/\hat{A}D)D}{T}u(t) = \frac{1}{\hat{A}}u(t)$$

This is the filtering used in identifying output error models, so in the limit as N_2 tends to infinity the identification scheme tends to output error form, even though the predictive model is an equation error model.

Kwok (1990) observed that when $D(q^{-1})$ contains Δ as a factor (as in most process control applications), then the E_j polynomials do not converge. In such

cases the identification method does not become output error LS as N_2 tends to infinity. Further details of the behaviour of L as N_2 becomes large may be found in Kwok (1991).

Observation 3 : *LRPI and LS are equivalent when identifying pulse response type models.*

Pulse response type models, such as the step response model used for DMC (Dynamic Matrix Control):

$$y(t) = \sum_{i=1}^{\infty} g_i \Delta u(t-i) + e(t)$$

contain purely numerator terms. There is no denominator in the process model and no $E_j(q^{-1})$ polynomial. L will therefore be equal to 1.0 when LRPI is used to identify this kind of model.

Observation 4 : $L(q^{-1})$ *is a weak function of the estimated* $\hat{A}(q^{-1})$ *and a strong function of* N_2*, the prediction horizon.*

Figure 5.5 shows the variations of the frequency response of $L(q^{-1})$ and $1/\hat{A}$ (a first order polynomial) as the pole of the process model transfer function (the zero of $\hat{A}$) varies from 0.8 to 1.0. The normalized cut-off frequency of the filter varies from 0.25 radians to 0.32 radians. The range of $\hat{A}$ polynomials shown in Figure 5.5 corresponds to process time constants between 4.48 samples and infinity. Obviously, the L filter is a weak function of the process time constant. The normalized cut-off frequencies of the corresponding $1/\hat{A}$ cover a relatively much broader range of 0.03 to 0.15 radians.

Fig.5.6 shows the variation of the frequency response of $L(q^{-1})$ with N_2 as it varies from 2 to 50. The normalized cut-off frequency varies from 2 radians to 0.15 and the maximum attenuation goes from 0.5 to 0.08. Clearly the prediction horizon N_2 plays a much stronger role in the nature of L than the pole of the model does.

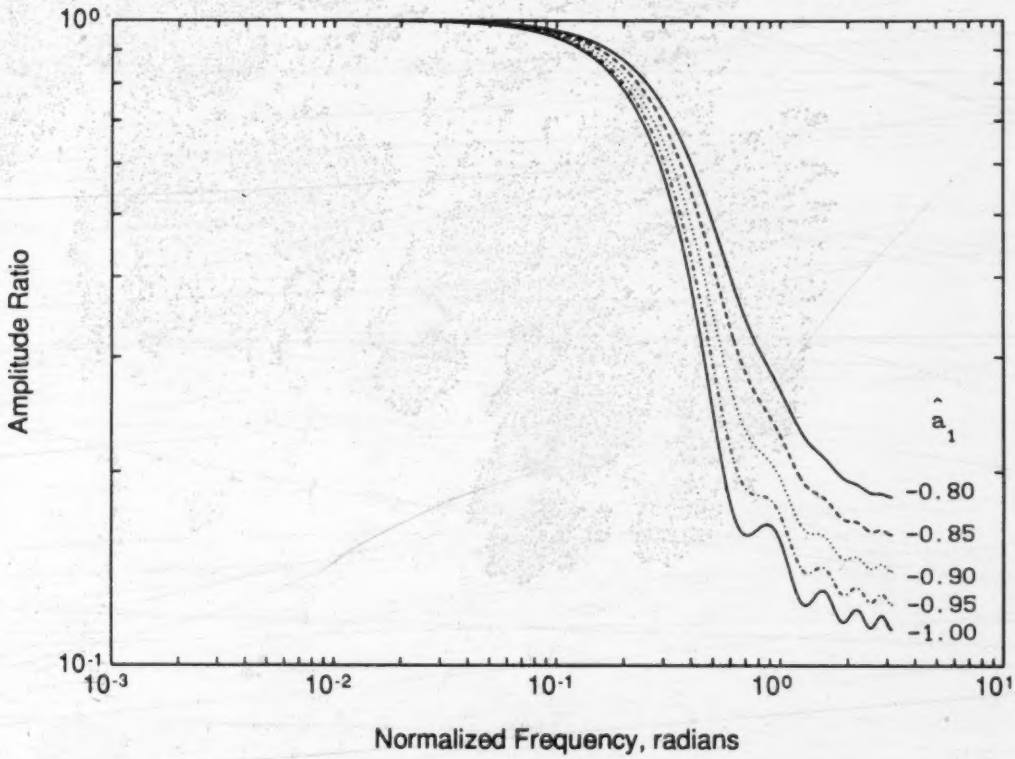

Figure 5.5: Variation of the Frequency Response of L with Process Model Pole Position

Observation 5 : *L is typically low-pass but under certain conditions it may become band-pass or even high pass. These conditions are rarely, if ever, encountered in chemical processes.*

If $\hat{A}$ (at least second order) is sufficiently lightly damped then L will show a resonant peak. At very slow sample rates the resonant peak will approach or even pass the Nyquist frequency, and then L will become a high-pass filter. In chemical engineering processes, the dominant time constant is usually many times larger than the control interval, so a high-pass form of L is unlikely. In mechanical systems, such as robots, very fast, very oscillatory responses are quite common.

The implementation of LRPI as LS with an additional time-varying (model- and tuning-dependent) prefilter provides insight into what has been happening in adaptive control research to date. Many researchers have recommended use of band-pass filters in identification. The justification has been that the model must fit the process best in the region within the passband. While this is true, use of this *ad hoc* filtering is an admission that the model cannot fully describe the plant. The result has been that the parameter estimator is identifying a model different from that which is being controlled, an apparent inconsistency. Although LRPI cannot completely remove the need for an *ad hoc* noise model, it does reduce the number of design choices required of the user.

Much of the discussion about *ad hoc* filtering has centered about matching the process "near" the crossover frequency (the frequency where the phase lag is equal to $-180°$. This of course means that the process crossover frequency must be known, at least within bounds. LRPI does not suffer from this requirement, as the L filter is a function of the process model: as the process model changes, so does the L filter. McIntosh *et al.* (1990) in particular recommend the use of an additional degree of filtering in the identification, but with weak justification. Their *ad hoc* filter was $1/(1-0.8q^{-1})$. It is shown in Figure 5.7 compared to the LRPI L filter for

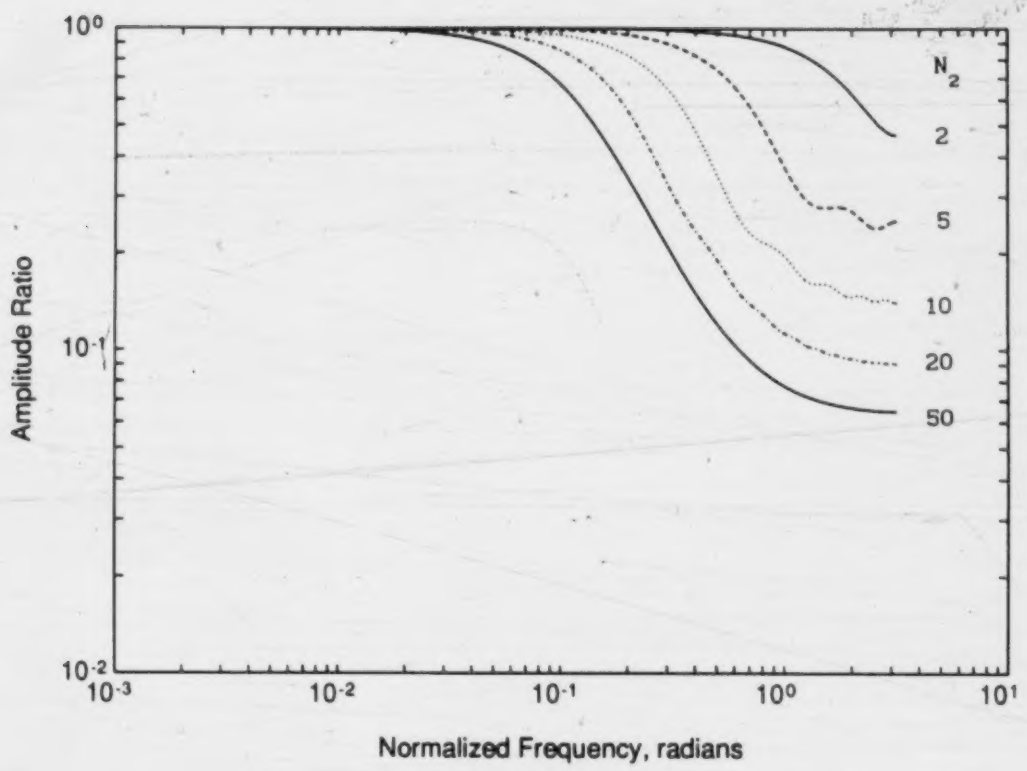

Figure 5.6: Variation of the Frequency Response of L with N_2.

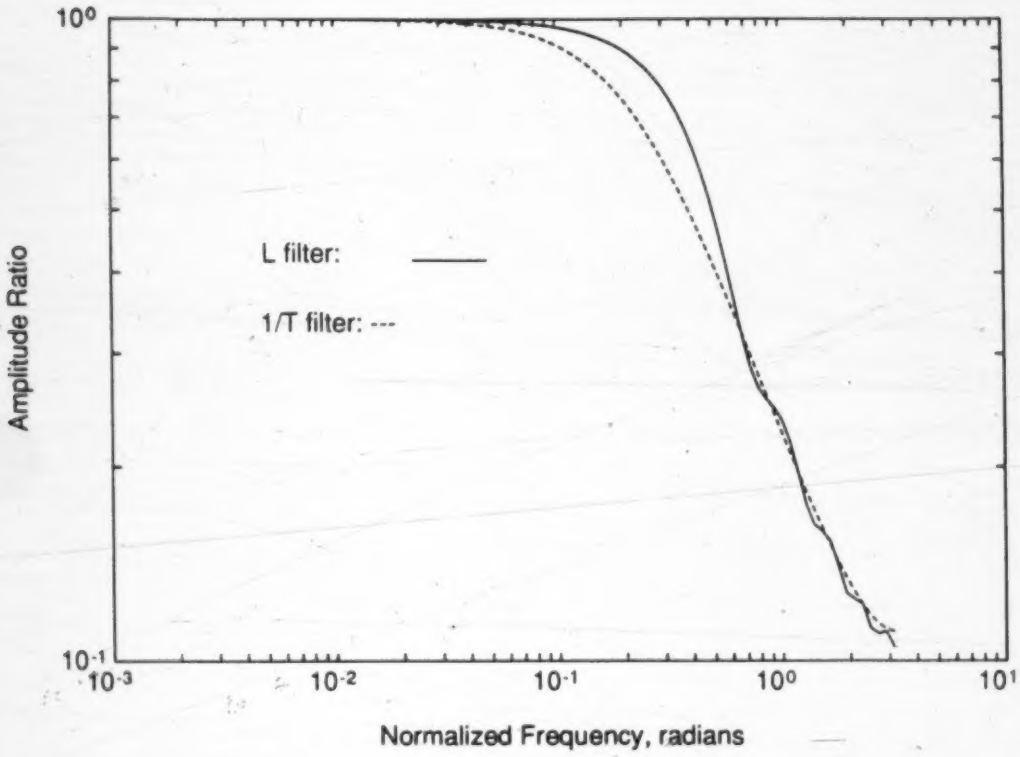

Figure 5.7: Comparison of L and *ad hoc* Filters

the "default" GPC tuning ($N_1 = 1,\ N_2 = 10,\ NU = 1,\ \lambda = 0,\ T = 1 - 0.8q^{-1}$) and a process model with $\hat{A} = 1 - 0.9q^{-1}$.

Traditionally, *ad hoc* prefiltering is performed using an infinite pulse response (autoregressive) filter. It would be instructive to examine the characteristics of approximating $L(q^{-1})$ with $1/T(q^{-1})$ with a view to assessing the prefilters typically chosen by the practitioners in the field. This can be easily done via a simple least squares solution to the equation:

$$L(q^{-1})T_{eq}(q^{-1}) \approx 1$$

Fig.5.8 shows the variation of the zero of a first order T_{eq} (the pole of the filter $1/T_{eq}$) with N_2. $\hat{A}$ is fixed at $1 - 0.9q^{-1}$. Note that the typical value of $T_{eq} = 1 - 0.8q^{-1}$ used by E. Lambert (1987) and McIntosh *et al.* (1990) is very close to the zero position

obtained here (i.e. 0.77 for $N_2 = 10$). This formalizes the choice of the estimator prefilters which were found more on a trial and error basis previously.

Note that the prefilter $L(q^{-1})$ is used only for the identification scheme to be able to mimic the effect of the longer range predictions with a single prediction estimator; it is *not* used as the noise model in the controller.

5.3.4 Stochastic Extensions

All of the discussion to date has been centred on the deterministic case. When there is actual noise (measurement noise or disturbances) the mathematics become somewhat more complicated, but the overall conclusions are the same. As pointed out earlier, it is common to model disturbances as filtered white noise. In our example, this implies that:

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + \frac{T(q^{-1})\xi(t)}{D(q^{-1})}$$

In most applications $D = 1 - q^{-1}$. In cases where there are cyclic disturbances (e.g. eccentricity control in rolling sheet metal) $D(q^{-1}) = 1 - 2\cos\omega_0 h q^{-1} + q^{-2}$ where ω_0 is known *a priori*. It is sometimes possible to estimate this internal model of the disturbance on line (Ishitobi *et al.*, 1989). Estimation of $T(q^{-1})$ on line is generally not too successful. Here we assume that the transfer function T/D is known exactly. This admittedly unrealistic assumption is required to perform the following analysis.

Consider the system and model as above:

$$\begin{aligned} A^0(q^{-1})y(t) &= B^0(q^{-1})u(t-1) + \frac{T(q^{-1})}{D(q^{-1})}\xi(t) \\ \hat{A}(q^{-1})y(t) &= \hat{B}(q^{-1})u(t-1) + \frac{T(q^{-1})}{D(q^{-1})}\epsilon(t) \end{aligned}$$

Recall that:

$$\hat{y}(t+j|t) = \frac{E_j\hat{B}Du(t+j-1) + F_j y(t)}{T} \tag{5.33}$$

$$T = E_j\hat{A}D + q^{-j}F_j \tag{5.34}$$

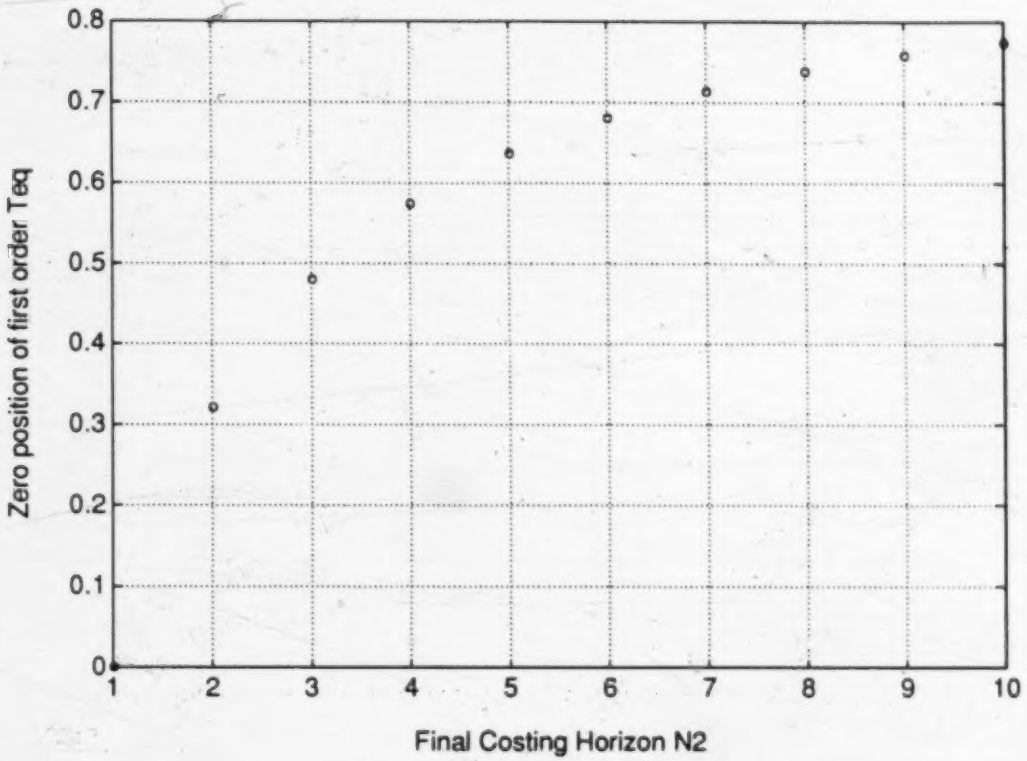

Figure 5.8: Variation of the Zero of T_{eq} with N_2.

Combining above relationships gives:

$$\begin{aligned}
\hat{y}(t+j|t) - y(t+j) &= \frac{E_j \hat{B} D u(t+j-1) + F_j y(t)}{T} - \left(\frac{B^0}{A^0} u(t+j-1) + \frac{T}{A^0 D} \xi(t+j) \right) \\
&= \frac{E_j \hat{B} D u(t+j-1) + F_j (\frac{B^0}{A^0} u(t-1) + \frac{T}{A^0 D} \xi(t))}{T} \\
&\quad - \left(\frac{B^0}{A^0} u(t+j-1) + \frac{T}{A^0 D} \xi(t+j) \right) \\
&= \frac{E_j \hat{A} D}{T} \left[\left(\frac{\hat{B}}{\hat{A}} - \frac{B^0}{A^0} \right) u(t+j-1) - \frac{T}{A^0 D} \xi(t+j) \right]
\end{aligned}$$

For open loop estimation $u(t)$ is *uncorrelated* with $\xi(t)$. It is then straight forward to see that the cost minimized by LRPI, i.e.

$$J_{LRPI} = \mathcal{E} \left\{ \sum_{j=N_1}^{N_2} (\hat{y}(t+j|t) - y(t+j))^2 \right\}$$

where $\mathcal{E}$ is the expectation operator, can be rewritten as:

$$J_{LRPI} = \frac{h}{2\pi} \int_{-\pi/h}^{\pi/h} \sum_{j=N_1}^{N_2} \left| \frac{E_j \hat{A} D}{T} \right|^2 \left\{ \left| \frac{\hat{B}}{\hat{A}} - \frac{B^0}{A^0} \right|^2 \Phi_u + \left| \frac{T}{AD} \right|^2 \sigma^2 \right\} d\omega$$

Note that the above expression is only valid for open loop estimation where $u(t)$ and $\xi(t)$ are uncorrelated. The assumption regarding the exact knowledge of T/D ensures convergence of the parameters to their exact values in the case of correct parameterization and sufficient excitation in the input for open-loop estimation.

5.3.5 On-line LRPI Implementation

On-line implementation of adaptive GPC using LRPI is shown diagrammatically in Figure 5.9. Each time new measurements are received, they are filtered using the previously calculated L filter, along with the usual $\Delta/T(q^{-1})$. The filtered measurements are then used to update the process model which is then used both for control purposes and to update the L filter.

There are two slight variations in application, depending on whether the identification and control tasks share information or are implemented separately. The

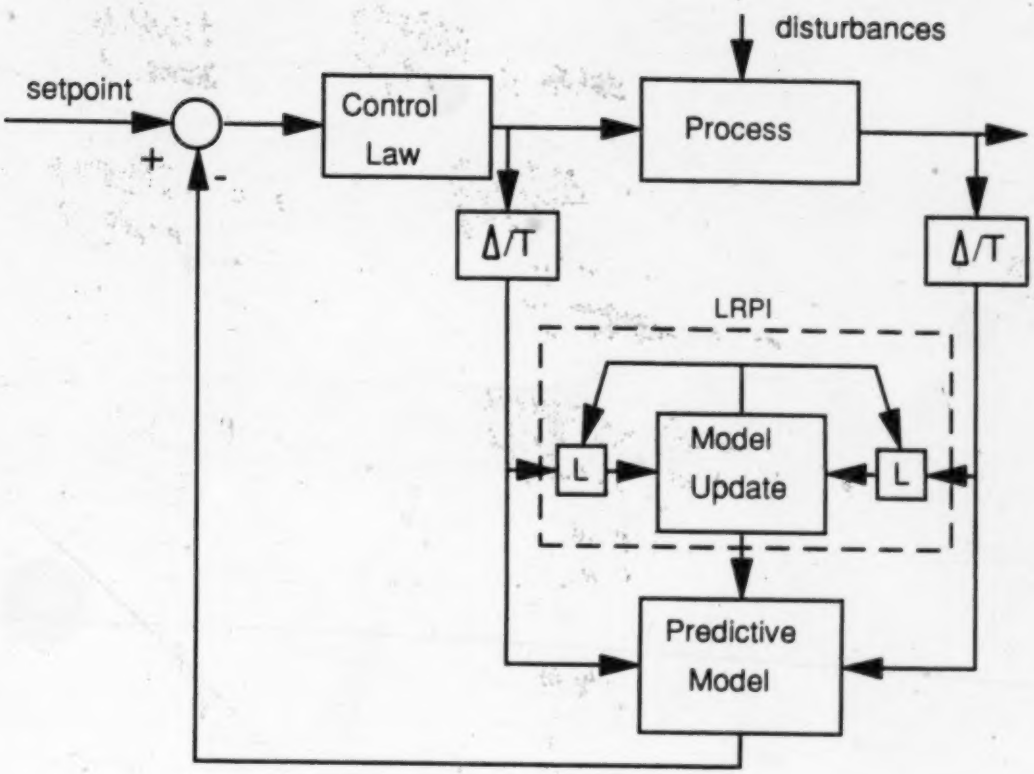

Figure 5.9: On-Line Implementation of Adaptive GPC with LRPI

only difference is the calculation of the $E_j(q^{-1})$ polynomials for LRPI. If the two routines share enough information, the E_j polynomials from the controller may be passed to the estimation and used for the next filter calculation, otherwise LRPI will have to repeat the Diophantine equation calculations.

Chapter 6

Evaluation of LRPI

In this chapter LRPI is evaluated through simulation and experimental studies. It will be compared to LS with and without the *ad hoc* prefilter. The best test of the newly proposed algorithm is through simulation and experimental evaluation. This is the main objective in the present chapter. While only application-specific conclusions can be drawn, the evaluation nevertheless serves to illustrate the practicality of the newly proposed algorithm.

6.1 Simulation Examples

LRPI was examined using a standard benchmark simulation: the plant of Rohrs *et al.* (1984). This plant has been used in a number of studies to illustrate the stability problems in adaptive closed loop systems in the presence of model-plant mismatch. This simulated plant is a third order process with a dominant first order response and high frequency underdamped second order dynamics. For the purposes of this work, the third order plant was modelled as first order, thus requiring some treatment or compensation for unmodelled dynamics. The actual plant is given by:

$$G(s) = \frac{2}{s+1}\frac{229}{s^2+30s+229} \qquad (6.1)$$

and time is in seconds.

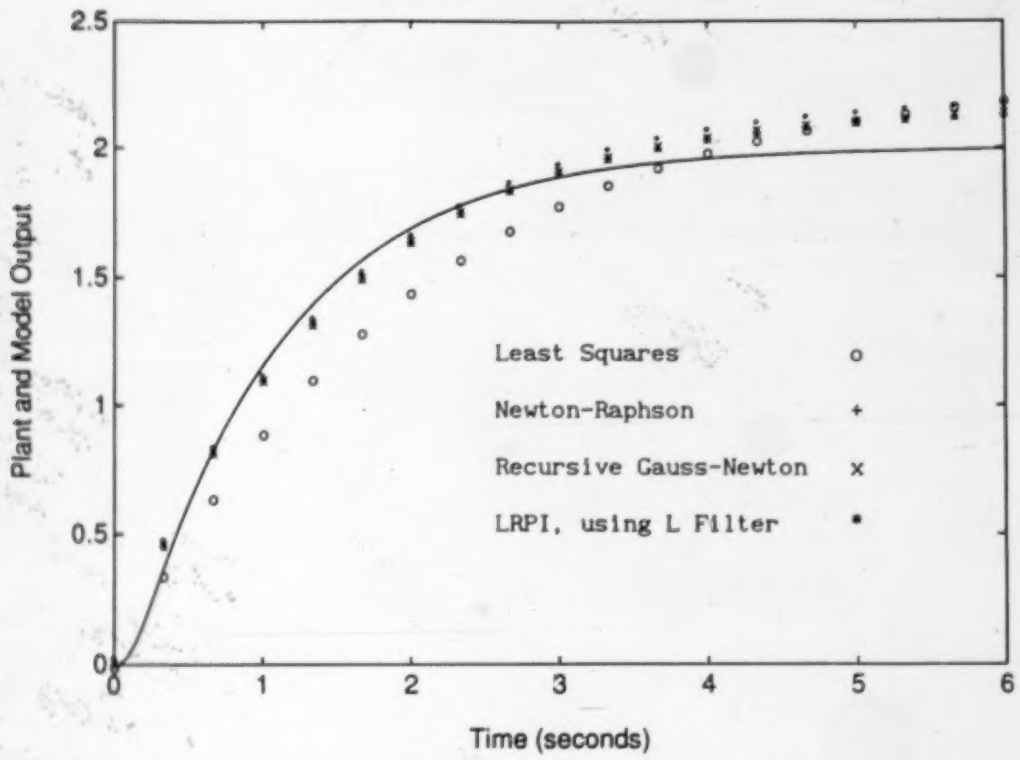

Figure 6.1: Step Response of Rohrs *et al.* Plant and First Order Models for Sampling Frequency of 3 Hz

6.1.1 Open Loop

The process was simulated for open-loop identification initially with a sample frequency of 3 Hz. The input sequence was $N(0,1)$ for 1000 samples. The process step response is shown by the solid line in Figure 6.1. Note the slightly sigmoidal shape: the slope is initially zero and increases for the first fraction of a second. For slow sampling (3 Hz) it is possible for a first order least squares model to describe the process reasonably, as can be seen from the circles in the same figure.

Three different methods were used to calculate the LRPI model for 3 Hz sampling: Newton-Raphson (batch analysis), recursive Gauss-Newton and the L filter with RLS. The step responses of resulting models are shown by the other sets of symbols in Figure 6.1. While there are differences among the three models, they

are minor. The differences are caused by the approximations inherent in the recursive methods, but there is no significant effect on the model.

For faster sampling the quality of the LS model declines. For a sample rate of 20 Hz, the LS model sacrifices all low-frequency accuracy for matching the one-step-ahead prediction. This is shown by the circles in Figure 6.2. Not surprisingly, the GPC controller calculated using this model is closed loop unstable. LRPI was used to find process models for $N_1 = 1$ and $N_2 = 2, 5, 10$. Their step responses are also shown in Figure 6.2. Although no first order model can exactly match the process at this high frequency, the LRPI model for $N_2 = 10$ obviously provides a better match over the range from 1 to 10 steps than the LS model does. The closed loop poles for the process controlled by GPC using the different models are shown in Figure 6.3. It should be clear from Figure 6.3 that the better models give better closed loop control.

6.1.2 Closed Loop

Adaptive control of the third order simulation of Rohrs' plant was carried out to examine the effect of LRPI in closed loop. Four different runs were performed:

1. Full order GPC with standard RLS,
2. GPC with first order model, noise model (data pre-filter) $T = 1 - 0.9q^{-1}$, and standard RLS,
3. GPC with first order model, noise model $T = (1 - 0.8q^{-1})^2$, and standard RLS,
4. GPC with first order model, noise model $T = 1 - 0.8q^{-1}$, and LRPI.

For all simulations the sample rate was 20 Hz. The setpoint was a square wave between 0 and 1 with a 5 second period. The initial covariance matrix was $10\mathbf{I}$ and $\hat{b}_1$ was initialized to 1 with all other parameters set to 0. The GPC tuning parameters were: $N_1 = 1$, $N_2 = 20$, $NU = 1$, $\lambda = 0$.

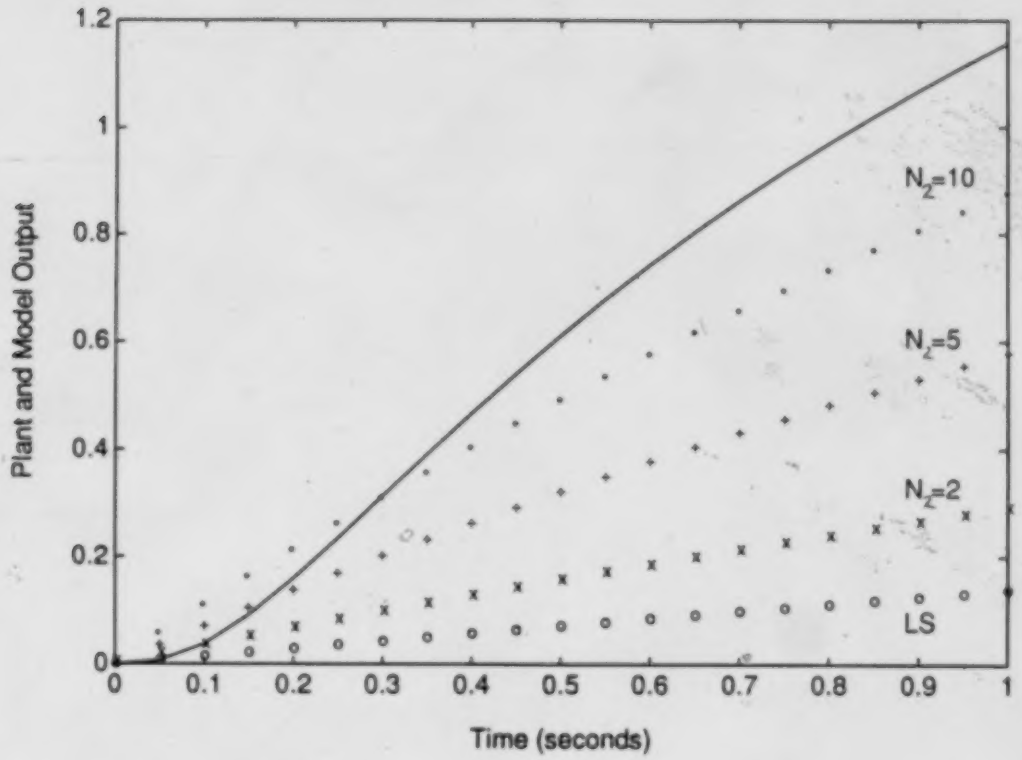

Figure 6.2: Step Response of Rohrs *et al.* Plant and First Order Models for Sampling Frequency of 20 Hz

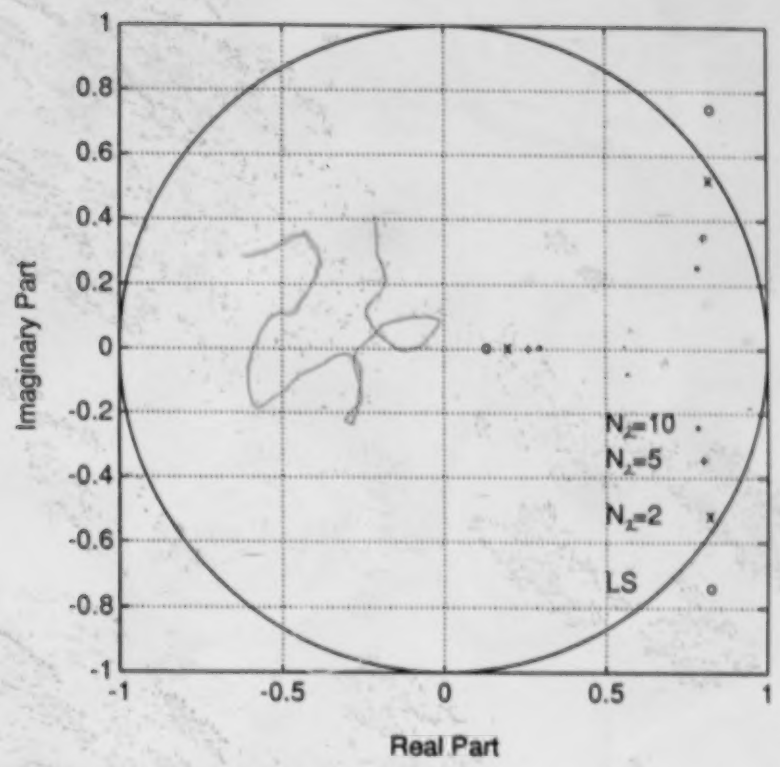

Figure 6.3: Closed Loop Poles for GPC and LRPI with Different Values of N_2

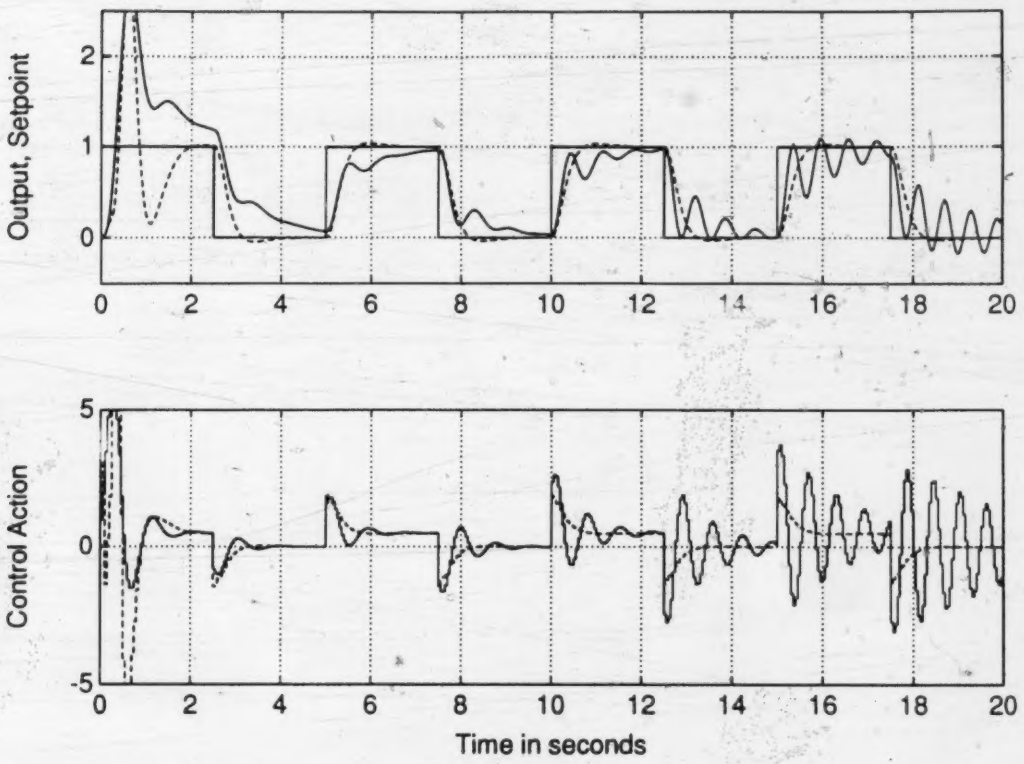

Figure 6.4: Response of Self-tuning GPC for First Order Model, $T = 1 - 0.9q^{-1}$ and Standard RLS

Since the best possible control is achieved using the full order model, all other runs are compared to run #1. The dashed line in Figures 6.4, 6.5 and 6.6 is the full order model case, i.e. there is no structural model-plant mismatch and the parameters quickly converge to the correct values.

Figure 6.4 shows the response when GPC is used in conjunction with RLS with a first order T filter and no additional filtering for identification. Note the highly oscillatory response. Moreover, the oscillations are becoming worse. Oscillatory response is fairly common for control with reduced order models. The oscillations can be reduced by increasing the filtering, as seen in Figure 6.5. The oscillations have been damped and control is now stable, but the response is quite detuned. The price

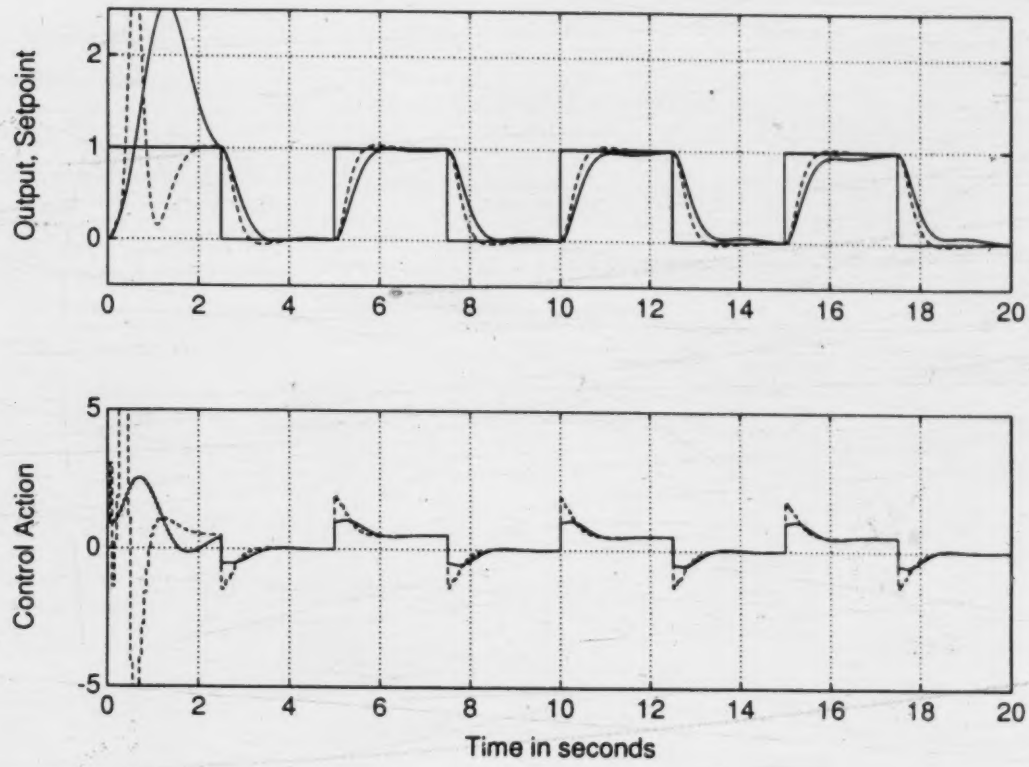

Figure 6.5: Response of Self-tuning GPC for First Order Model, $T = (1 - 0.8q^{-1})^2$ and Standard RLS

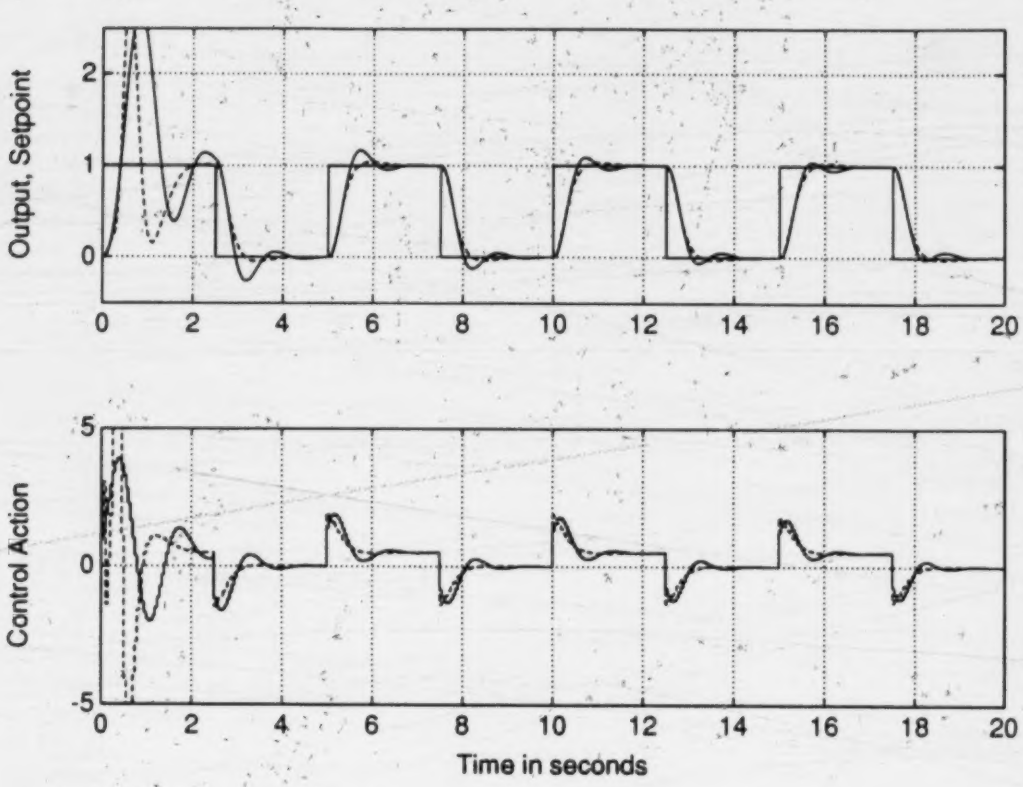

Figure 6.6: Response of Self-tuning GPC for First Order Model, $T = 1 - 0.8q^{-1}$ and LRPI

for better modelling is a detuned controller. In contrast, the use of LRPI improves the modelling but without slowing the controller unacceptably (Figure 6.6). LRPI therefore shows some promise for experimental application.

6.2 Experimental Studies

Experimental studies were carried out on a pilot-scale stirred tank heater in the Chemical-Mineral Engineering Building at the University of Alberta. The equipment is illustrated in Figure 6.7. It consists of a double-walled glass tank 50 cm high with an inside diameter of 14.5 cm. Cold water enters the tank and is heated by a steam coil. The water leaving the tank passes through a long 1 inch copper pipe with four thermocouples placed along its length. The different thermocouples provide a choice of transport delays. For this evaluation a dead-time dominated system was chosen, with dead-time approximately equal to one time constant. The water temperature was controlled using the steam valve position, and the disturbance was the inlet water flowrate. An IBM PS/2 Model 70 computer operating under the QNX operating system performed the control calculations, using an Opto-22 Multiplexer for A/D and D/A conversions. The QNX operating system provides a real-time, multitasking environment suitable for evaluating control strategies. The controller ran within Multicon, a software package developed at the University of Alberta for testing and evaluating different identification and control programs. Multicon provides on-line access to controller tuning parameters as well as a facility for producing trend plots of selected variables.

The inlet water flow rate was set at the start of each experiment to 36 cm^3/s. The temperature was measured a significant distance downstream from the tank, and the resulting dead time was 36 seconds. Since the sample time was 8 seconds and the assumed dead time was five samples (plus zero-order hold), a reasonable degree of structural model-plant mismatch was guaranteed.

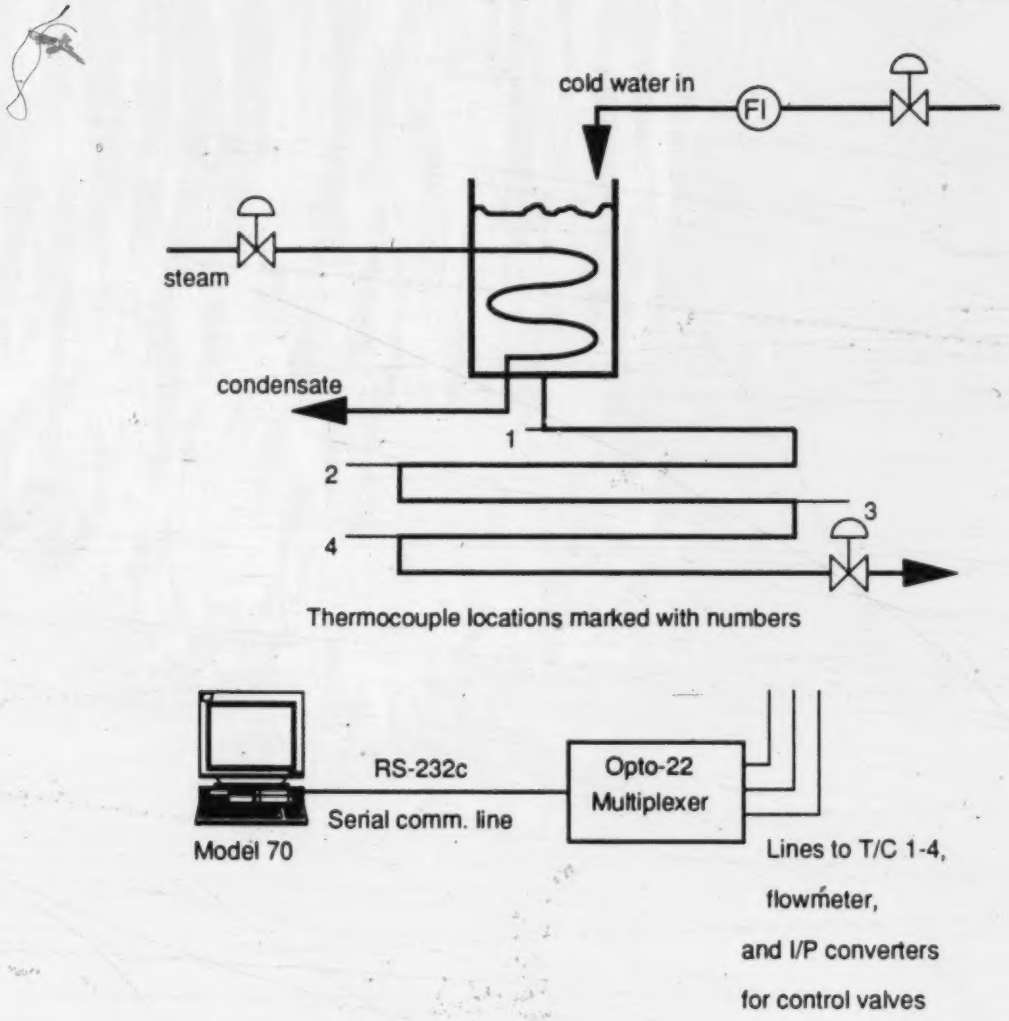

Figure 6.7: Experimental Equipment

The GPC (and LRPI) prediction horizons N_1 and N_2 were set to constant values of 6 and 15, respectively. All experiments were carried out with the same pattern of disturbance and setpoint changes. In all cases the GPC controller was switched into automatic mode after an initial tuning period of 25 samples. During the tuning period the steam valve (the final control element) was moved in a doublet 5% either side of the initial position of 30% open. The temperature setpoint was raised 5°C after 50 samples and then returned to its initial value following another 50 samples. Later in the experiment, at 400 samples, the setpoint was raised 5°C again for another 300 samples. The inlet water flow rate was increased to 45 cm^3/s after 200 samples and then returned to 36 cm^3/s 100 samples later. The disturbance was repeated from 500 to 600 samples. These flow changes acted both as disturbances and changes in process (in terms of time constant and time-delay changes), thus presenting the controller with a significant challenge. The controller was forced not only to cope with the disturbance, but also to deal with the fact that its model parameters were out of date.

A least squares estimator with a variable forgetting factor was used for implementing LRPI. The directional forgetting factor of Kulhavý (1987) was used to discount old data, since it was known that the best plant model would change during the experiment, and the main purpose of the directional forgetting factor is to deal with randomly varying process parameters

Four adaptive control configurations were chosen. For three of them a first order process model was used. The GPC controller T filter was fixed to $1 - 0.8q^{-1}$ and three different identification filtering options were evaluated: RLS with a first order T filter matching that of the controller, RLS with a second order T filter of $5(1 - 0.8q^{-1})^2$ (the factor of 5 was used to keep the steady state gain of the second filter equal to unity) and LRPI. The fourth controller used LRPI and a second order process model with five numerator parameters. Naturally, the data filtering for LRPI included the effects of the Δ and T filter of the controller disturbance model.

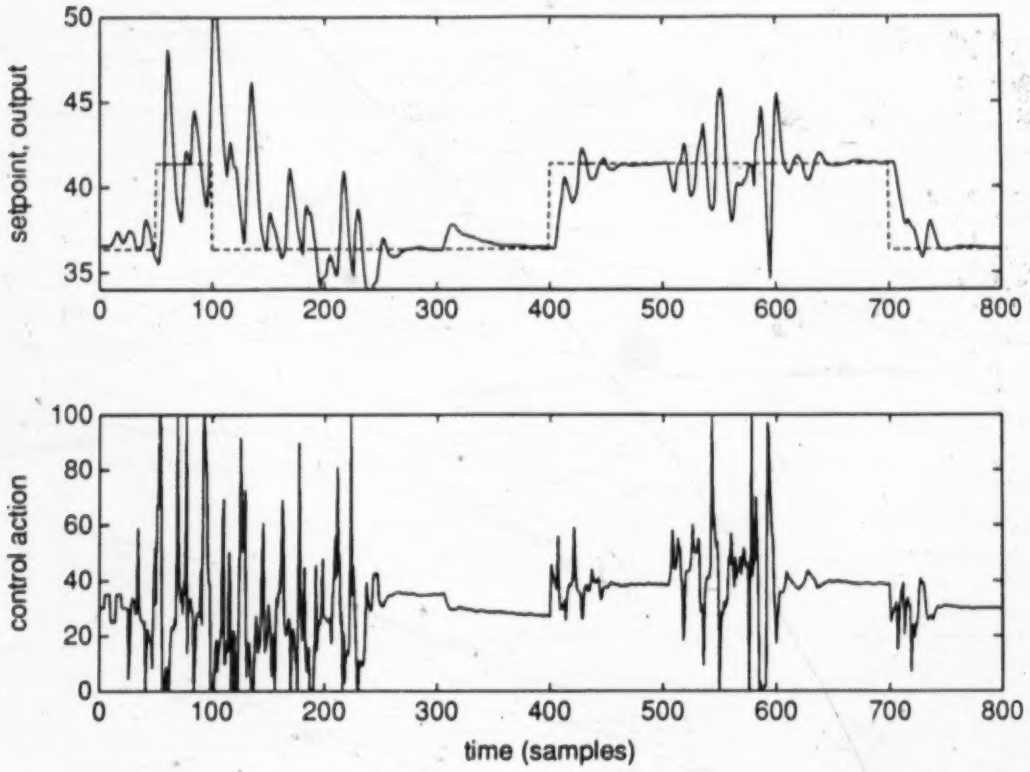

Figure 6.8: Process Input and Output for GPC with LS Estimator.

The results of the experimental trials are shown in Figures 6.8, 6.9 and 6.10. Control with a first order T filter applied to the estimation is obviously unacceptable. Use of a second order T filter results in significantly better control, very similar to the LRPI results. There are large excursions when the disturbance enters the process because of the long dead time and relatively detuned controller.

The large fluctuations in the control signal and parameter estimates in the first case, shown in Figure 6.11, are caused by the emphasis placed on the high frequency part of the prediction error due to insufficient signal filtering for identification. The use of a variable forgetting factor results in large parameter fluctuations, but is required if the "true" parameters are known to vary (Kulhavý and Karny, 1984).

The model parameters for GPC control with the *ad hoc* filter and the LRPI

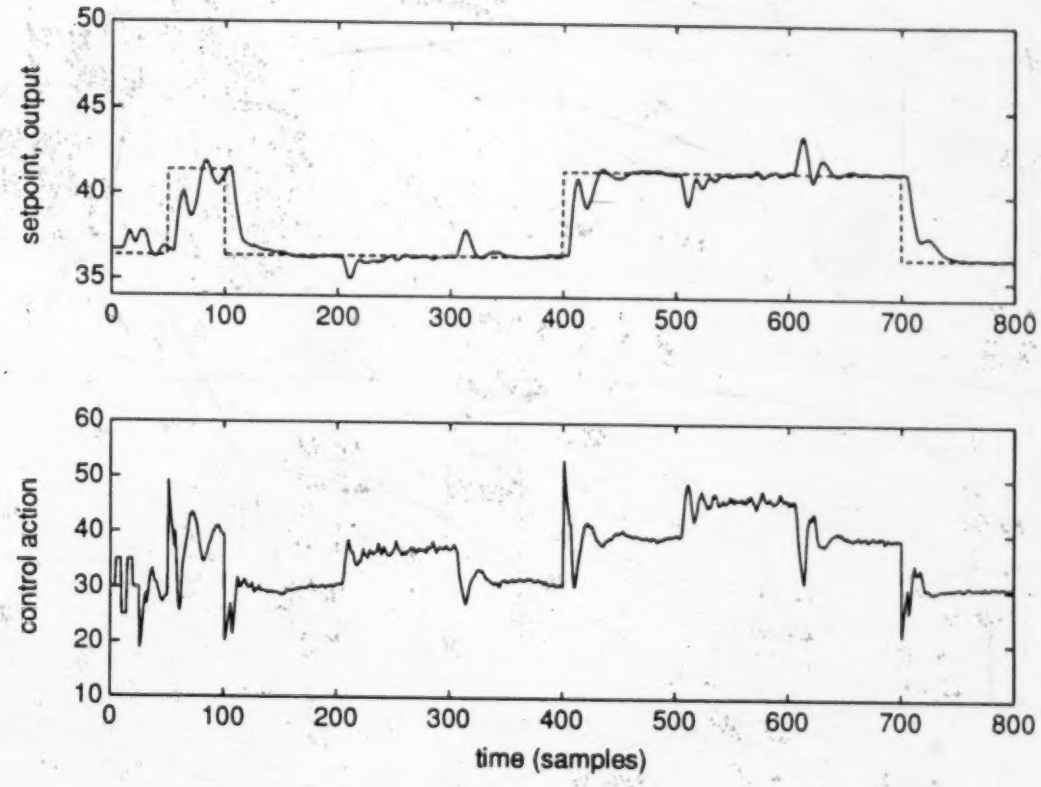

Figure 6.9: Process Input and Output for GPC with LS Estimator and an *ad hoc* Filter

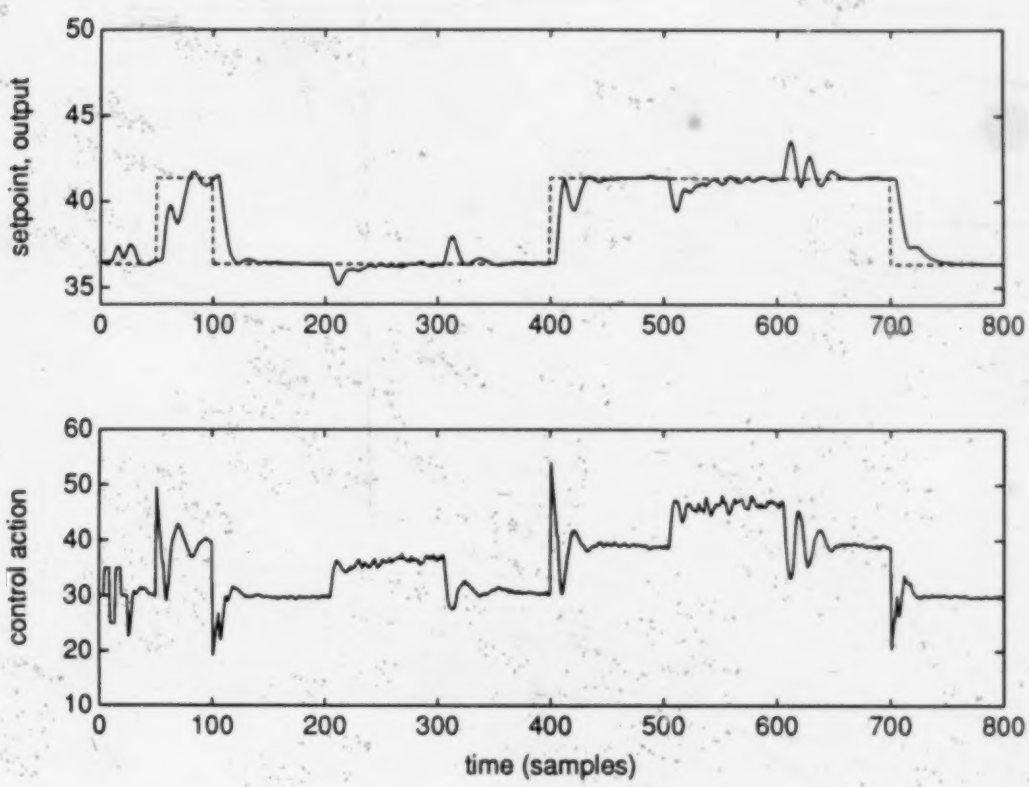

Figure 6.10: Process Input and Output for GPC and LRPI

filter are shown in Figures 6.12 and 6.13 and show no such behaviour. It is remarkable how similar the process temperature and parameter trajectories are for these two cases. The only apparent difference is in the convergence rate: the LRPI parameters change more slowly than the LS parameters during the changes in process. The difference is caused by the difference between the L and T filters and is likely insignificant, since the closed loop control performance is almost identical.

Figure 6.14 shows the evolution of the LRPI filter during the run. The initial vlaue of L was a FIR approximation of $1/(1-0.8q^{-1})$. After eight samples the filter had essentially converged. Subsequent changes were in response to changes in process model parameters throughout the run. The *ad hoc* T filter is also shown in Figure 6.14, and it can be seen that the *ad hoc* filtering is very similar to the actual "optimal" L filter, for this particular model.

A fourth experiment was performed, using a second order model with an over-parameterized numerator to accommodate the varying dead time. LRPI was used as the identifcation method. The control performance was better than that achieved using a first order model, as can be seen in Figure 6.15, where the results are compared to the performance of GPC using the first order model and LRPI. Use of the high order model improves control notably during setpoint changes; however the disturbance rejection is improved less dramatically. This is because of the long process dead time: feedforward control is needed for any substantial improvement in disturbance rejection. The experiment using the high order model shows the improvement in control concomitant with a more sophisticated model: the improvement in control here does not seem to justify the extra calculations required.

6.3 Conclusions

In the presence of structural model-plant mismatch, LRPI permits the identification of models that are more useful for the purposes of GPC than those identified

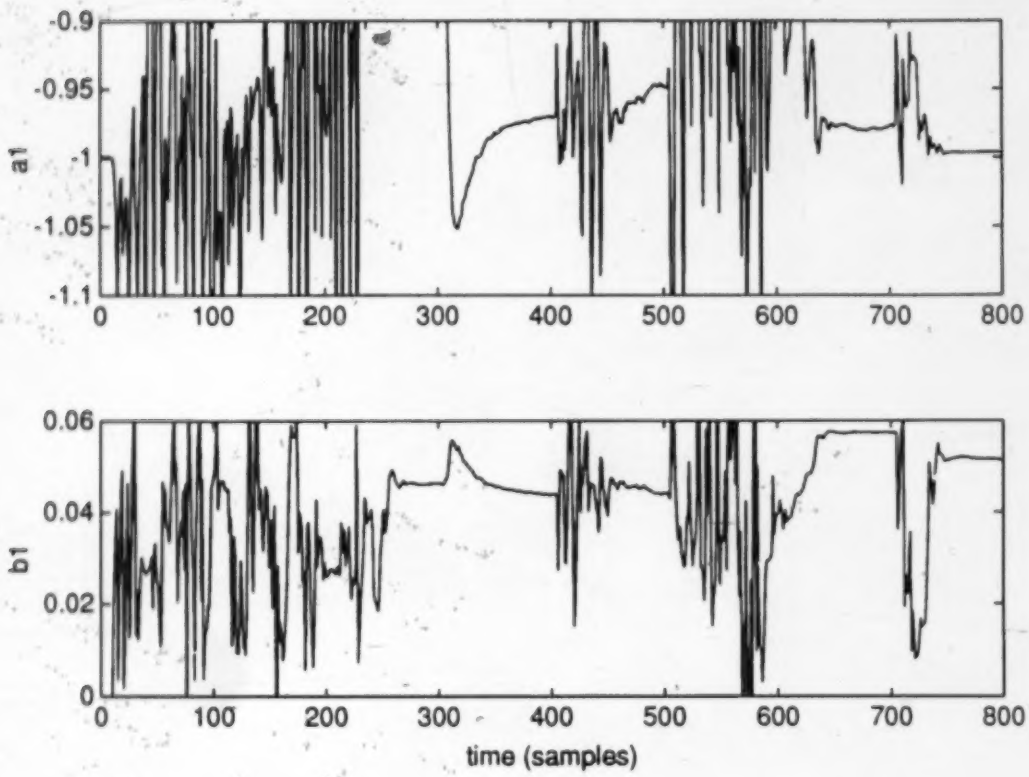

Figure 6.11: Parameter Trajectories for GPC with LS Estimator.

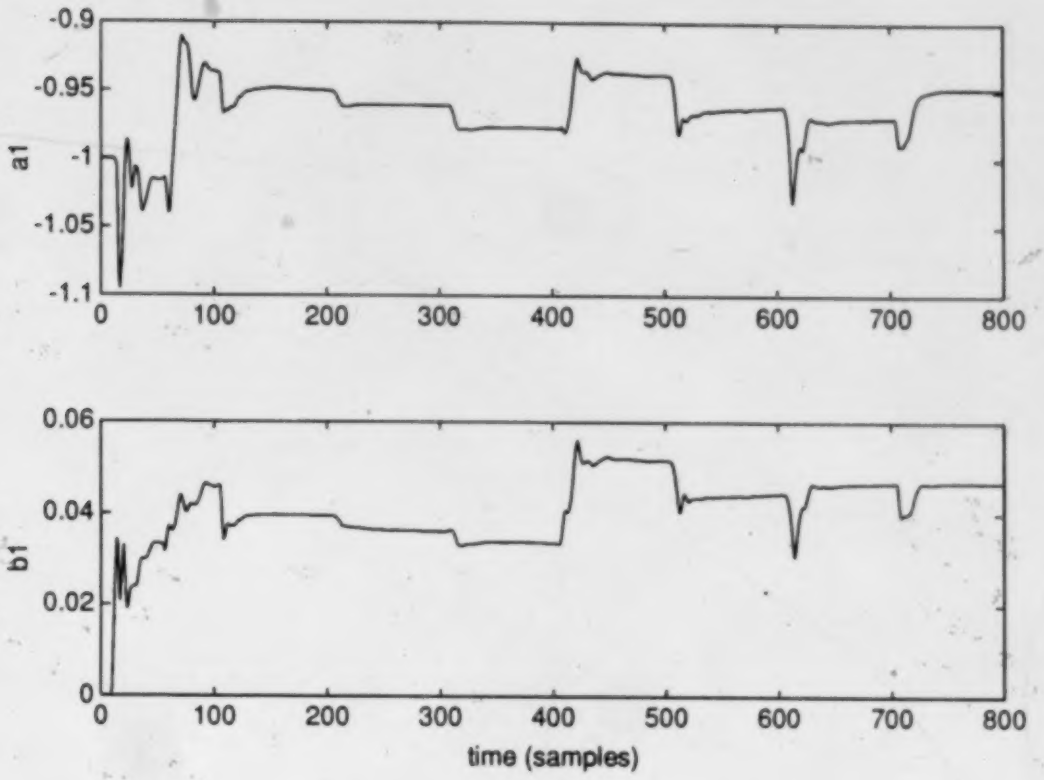

Figure 6.12: Parameter Trajectories for GPC with LS Estimator and an *ad hoc* Filter

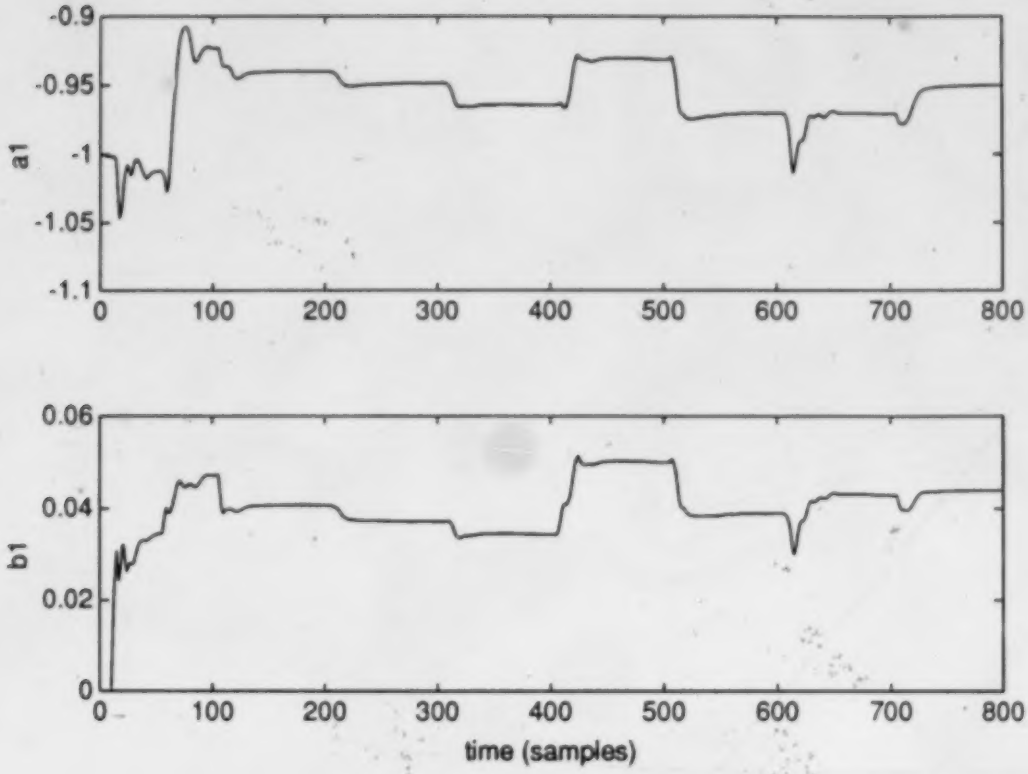

Figure 6.13: Parameter Trajectories for GPC and LRPI

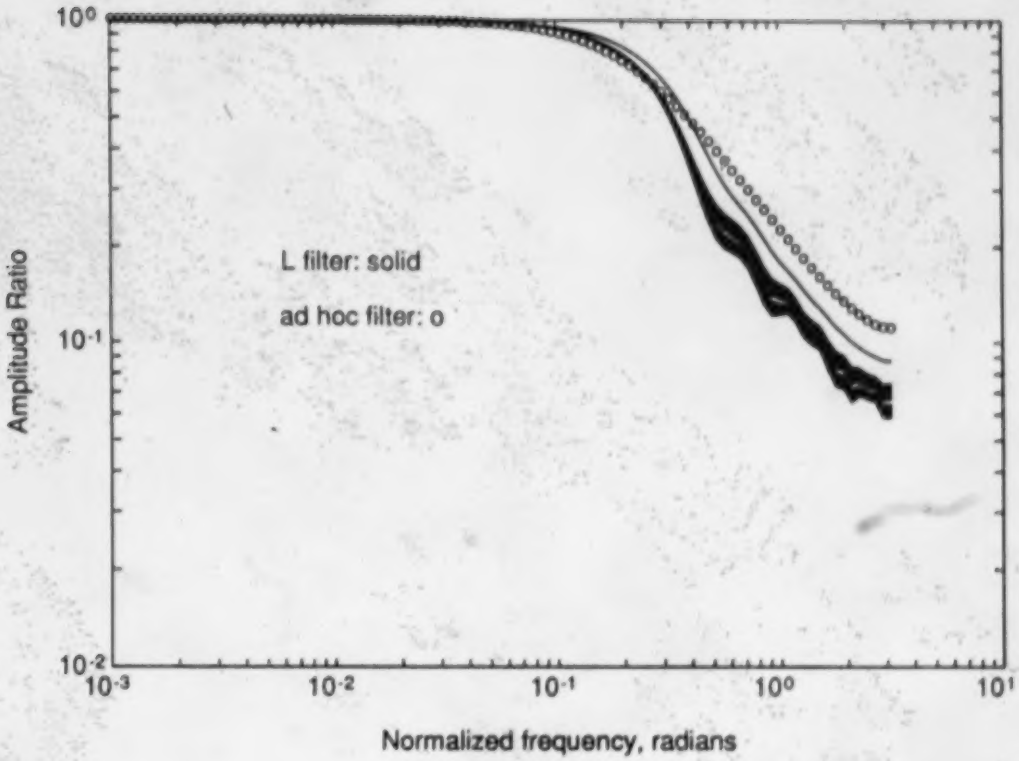

Figure 6.14: Evolution of LRPI Filter

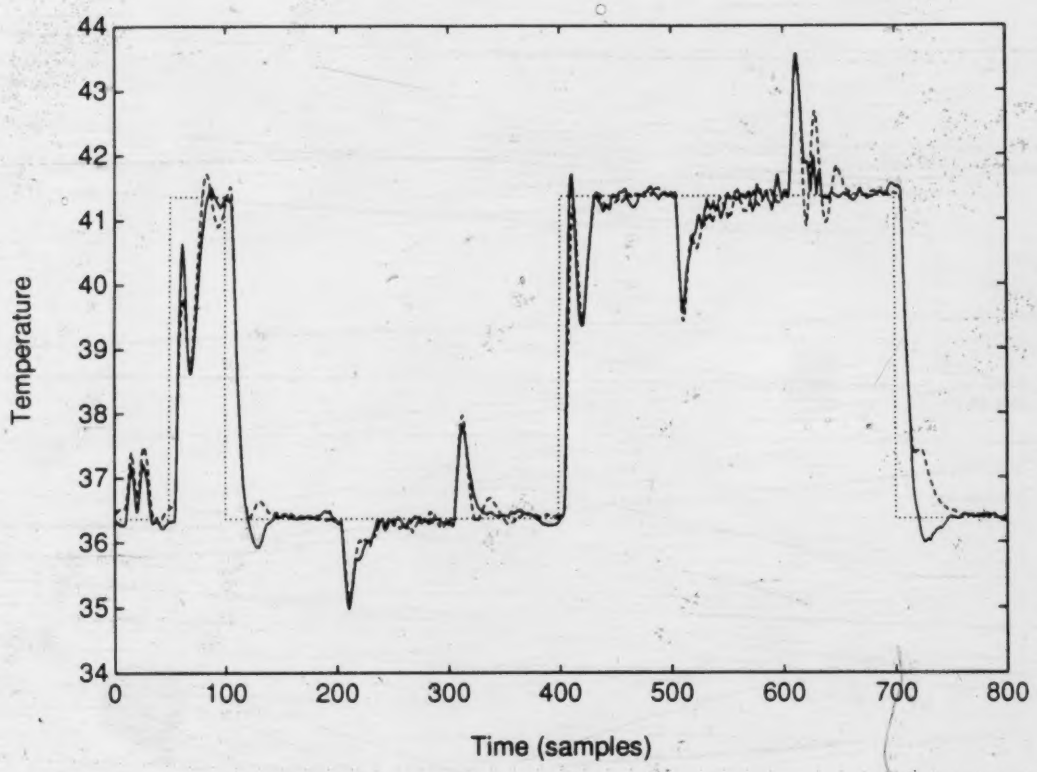

Figure 6.15: Comparison of GPC Using Low and High Order Models

using RLS. Adaptive GPC implementations using LRPI as the parameter estimation method provided better quality control than those using standard RLS when model-plant mismatch was present (Figures 6.8 and 6.10).

The performance of adaptive GPC with LRPI was very similar to the performance when RLS was used with an additional *ad hoc* data pre-filter (Figures 6.9 and 6.10). The *ad hoc* pre-filter was seen to have frequency-domain characteristics very similar to the LRPI L filter. The advantage of LRPI is that the pre-filter need not be chosen *a priori* by the user: the adaptation method finds the optimal pre-filter automatically. This is of limited utility for low-order models and fixed controller tuning, but in cases where N_1 or N_2 may be changed more frequently, or when a higher order $A(q^{-1})$ polynomial is used, it is extremely useful. In cases where on-line calculation of the L pre-filter is impossible, the user can design a pre-filter using a "nominal" model and controller tuning and compare it to the "optimal" L filter.

In conclusion, LRPI is a practical, useful, identification method for adaptive GPC implementations, providing high-quality control while reducing the amount of work required of the user: the *ad hoc* data pre-filter is removed.

Chapter 7

Nonminimal Predictive Control

An alternative approach to the overall control objective of Chapter 4 was formulated by Lu and Fisher (1990a,b). The Nonminimal Predictive Controller (NPC) uses a more sophisticated predictor structure to simplify the control-relevant identification problem. Rather than reworking the identification, the entire adaptive controller is redesigned to minimize the overall cost. As such it is a more elegant solution to the complete adaptive control problem. NPC is described in this chapter and a few simulations are carried out to compare its capabilities to GPC with LRPI.

7.1 Design Philosophy

NPC is a different controller than GPC (and is therefore beyond the original scope of this work) but it is so similar in design philosophy and implementation that it is discussed here in some detail.

As with GPC, the control objective is to minimize the sum of squares of future control errors over a finite-time horizon:

$$J_{AC} = \sum_{j=N_1}^{N_2} ((y_{sp} - y(t+j))^2$$

using NU control actions. Distinct from GPC is the way NPC forms the predictions.

The GPC predictions are formed through recursive solution of the Diophantine equation (equation 5.6):

$$T(q^{-1}) = E_j(q^{-1})\hat{A}(q^{-1})\Delta + q^{-j}F_j(q^{-1})$$

It has been shown (e.g. Mohtadi, 1986) that the GPC predictor is equivalent to calculating $\hat{y}(t+1|t)$ and then using that value to find $\hat{y}(t+2|t)$ and so on. This is the optimal minimal predictor, but in the presence of noise the prediction errors grow quickly as the prediction distance increases.

A different prediction structure is used in NPC. The philosophy of NPC is that all predictions of future plant outputs should be equally accurate. This laudable goal is called the *equal-variance principle*, stated here without proof.

Theorem 1 *Let $\hat{y}(t+j|t)$, $j = 1,2,\ldots,N_2$ be the predicted values of $y(t+j)$, $j = 1,2,\ldots,N_2$ produced at time instant t by a multistep prediction model given a model parameter vector θ. If the multistep prediction model structure satisfies*

$$\hat{y}(t+j|t) = \hat{y}(t+j|t+i-1)$$

for $j = 1,2,\ldots,N_2$, $i = 2,\ldots,j$ and $t \geq 0$ then, subject to the boundedness assumption of prediction errors, the time averages of the N_2 prediction error squares are the same, i.e.

$$\lim_{N\to\infty}\frac{1}{N}\sum_{t=0}^{N-1}\frac{1}{N_2}\sum_{j=1}^{N_2}[y(t+j)-\hat{y}(t+j|t)]^2 = \lim_{N\to\infty}\frac{1}{N}\sum_{t=0}^{N-1}[y(t+1)-\hat{y}(t+1|t)]^2$$

So NPC uses a predictive model that provides long range predictions that are every bit as good as the one step ahead predictions.

There are other controllers whose predictive models fit the equal-variance principle, notably DMC. The use of a step-response model in DMC removes the need for intermediate predictions and so provides a prediction equally good for all ranges. This is of course for the deterministic case. In the presence of noise, the

noise model is of paramount importance in providing good long range predictions, regardless of the prediction structure. It will be shown later that the NPC noise model is similar to that of GPC.

As an example, consider the GPC predictor, discussed at length in Chapters 4 and 5. The process model is ARIMAX in structure:

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + \frac{T(q^{-1})}{\Delta}\xi(t)$$

and the j-step ahead prediction is calculated using the Diophantine equation:

$$T(q^{-1}) = E_j(q^{-1})A(q^{-1})\Delta + q^{-j}F_j(q^{-1})$$

If the model is correct, then $y(t+j)$ may be written as follows:

$$y(t+j) = \frac{F_j y(t)}{T} + \frac{E_j B \Delta u(t+j-1)}{T} + E_j\xi(t+j) \tag{7.1}$$

The prediction, $\hat{y}(t+j|t)$, is given by

$$\hat{y}(t+j|t) = \frac{F_j y(t)}{T} + \frac{E_j B \Delta u(t+j-1)}{T} \tag{7.2}$$

so the j-step ahead prediction error is the last term in equation 7.1. The term is ignored in the prediction because its expected value is zero. Its variance, however, is given by the following formula:

$$\sigma^2_{\epsilon_j} = \sigma^2(y(t+j) - \hat{y}(t+j|t)) = \sum_{i=0}^{j-1} e^2_{j_i}\sigma^2_\xi \tag{7.3}$$

It can be seen that the prediction error variance increases with prediction range for GPC.

For DMC, however, a truncated infinite step response model is used:

$$y(t) = \sum_{i=0}^{\infty} g_i \Delta u(t-i) + \xi(t) \tag{7.4}$$

The j-step ahead prediction is therefore

$$\hat{y}(t+j|t) = \sum_{i=0}^{\infty} g_i \Delta u(t+j-i) \tag{7.5}$$

and the prediction error is simply $\xi(t+j)$. Since ξ is assumed to be zero-mean and stationary, the prediction error variance is σ_ξ^2, regardless of prediction distance. DMC meets the equal variance principle, and GPC does not.

The DMC and GPC model structures are not completely equivalent. They do, however, possess one important quality. Both structures allow the controller to provide offset-free control in the presence of step-type disturbances. A complete discussion of noise models is beyond the scope of this discussion, and the reader is directed to McIntosh (1988), which discusses the difference between the two model structures in some detail.

7.2 The NPC Predictor

The NPC predictor is an interesting result. It is in appearance similar to the DMC predictor, but the philosophy is more akin to GPC. It is based on the multi-rate predictor of Lu *et al.* (1990) and the multi-rate inferential controller of Lu (1989).

Lu (1989) showed that for any process model, $\hat{A}(q^{-1}), \hat{B}(q^{-1})$, (of order na and nb respectively) there exists a polynomial $P_{N_2}(q^{-1})$ of degree $(N_2-1)\times na$ such that

$$P_{N_2}(q^{-1})\hat{A}(q^{-1}) = \hat{A}_{N_2}(q^{-N_2}) = 1 + a_{N_2 1}q^{-N_2} + a_{N_2 2}q^{-2N_2} + \ldots + a_{N_2 na}q^{-na N_2} \quad (7.6)$$

and

$$P_{N_2}(q^{-1})\hat{B}(q^{-1}) = \hat{B}_{N_2}(q^{-1}) \quad (7.7)$$

where $\hat{A}_{N_2}(q^{-N_2})$ is a polynomial in q^{-N_2} and $\hat{B}_{N_2}(q^{-1})$ is of degree $nbb = nb + N_2 \times (na-1)$. P_{N_2} is not necessarily easy to find; it is a function of the roots of $\hat{A}(q^{-1})$, but it does exist and can be found.

The predictor makes use of this as follows. The DARMA (Deterministic ARMA) equation:

$$\hat{A}y(t) = \hat{B}u(t-1)$$

is multiplied through by $P_{N_2}(q^{-1})$ and q^{+N_2} and becomes

$$\hat{A}_{N_2}(q^{-N_2})y(t+N_2) = \hat{B}_{N_2}(q^{-1})u(t+N_2-1) \tag{7.8}$$

$$\begin{aligned} y(t+N_2) \;+\; & \hat{a}_{N_2 1}y(t) + \hat{a}_{N_2 2}y(t-N_2) + \ldots + \hat{a}_{N_2 na}y(t-(na-1)N_2) \\ = \; & \hat{b}_{N_2 1}u(t+N_2-1) + \ldots + \hat{b}_{N_2 nbb}u(t+N_2-nbb) \end{aligned} \tag{7.9}$$

Since $\hat{A}_{N_2}$ is a function of q^{-N_2}, the only values of y needed are: $y(t)$, $y(t-N_2)$, $y(t-2N_2)$ and so on. No future values of the plant output are required, even implicitly, for long range prediction. Many control action values are used, both future and past but they are either known or to be solved for, and therefore present no problem. For prediction N_2-1 steps ahead, the predictor is rewritten for $t+N_2-1$, so $y(t-1)$ and $y(t-N_2-1)$ are used. The prediction error variance is the same for N_2 steps ahead or 1 step ahead using this predictor.

This is fine as far as it goes, but there is an obvious objection: $y(t)$ is not used for the shorter distance predictions. For predicting $y(t+1)$, for instance, the most recent value of y used in the predictor (equation 7.8) is $y(t+1-N_2)$, not $y(t)$ itself. Thus the predictor does not use the most recent process information. In addition, the question of how the nonminimal predictor deals with noise seems to be open.

The two objections are in fact one. The presence of $y(t)$ is not needed in the predictor in the deterministic case: it is only used when there is a noise model. If there is no noise then the prediction can be excellent without using any measured outputs. "Noise" is used here in its broadest sense, including model-plant mismatch as well as disturbances. By comparison the deterministic DMC predictor makes no use of y at all; it is only included as part of the noise model. The noise model is discussed in the next section.

In practice $\hat{A}_{N_2}$ and $\hat{B}_{N_2}$ are identified directly rather than calculating them from $\hat{A}$ and $\hat{B}$. This is because of the difficulty of calculating $P_{N_2}(q^{-1})$ on line. The fundamental drawback to NPC is a result of this: many parameters must be identified: $(na \times N_2)+nb$. If $N_2 = 10$ and the process has complex dynamics, it is possible

for there to be 40 or so B_{N_2} parameters. In addition, there is no guarantee that the identified parameters at any given time actually satisfy conditions 7.6 and 7.7. If the conditions are *not* satisfied, that is, there is no common factor between $\hat{A}_{N_2}$ and $\hat{B}_{N_2}$, then it cannot be guaranteed that the predictor is in fact equal variance. So, not only can there be a considerable numerical penalty, but the theoretical optimality is at risk. There are *ad hoc* methods to reduce the number of parameters required, but they lose all theoretical optimality: the equal-variance condition is not guaranteed.

The major advantage of the NPC predictor is the fact that the identification scheme to give the best long range prediction also happens to be the best one step ahead identification scheme: least squares. No adaptive filtering is needed; no alternative forms need be investigated; all that is required is simple least squares.

One implementation drawback of NPC is that the maximum value of N_2 to be used must be decided while commissioning the controller. The direct identification of the predictor, rather than the model, means that the maximum prediction distance is fixed. Since this is a direct method, the parameters identified are not useful for any other predictor. Once identification has started, N_2 cannot be increased, although it may be reduced. This is unfortunate, because N_2 is a primary tuning parameter.

7.3 The NPC Noise Model

The disturbance rejection properties of any controller are dependent on the noise model. In the case of NPC, it is assumed that the process may be described by a model of the following type:

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + \frac{C(q^{-1})}{\Delta_{N_2}}\xi(t) \tag{7.10}$$

where $C(q^{-1})$ is of order nc and $\Delta_{N_2} = 1 - q^{-N_2}$. If $N_2 = 1$, then $\Delta_{N_2} = \Delta$ and this is no more than the standard ARIMAX model. For NPC however, N_2 is chosen by the user and is typically closer to 10 than 1. $C(q^{-1})$ is chosen in much the same

way as for GPC: it is a stable polynomial that gives a good compromise between robustness and disturbance rejection.

Given such a process, a nonminimal model can be derived in a way similar to the deterministic case:

$$A_{N_2}(q^{-N_2})\Delta_{N_2}y(t) = \bar{B}(q^{-1})\Delta u(t-1) + \bar{C}(q^{-1})\xi(t) \tag{7.11}$$

where

$$\bar{B}(q^{-1})\Delta = B_{N_2}\Delta_{N_2} \tag{7.12}$$

$$\bar{B}(q^{-1}) = B_{N_2}(q^{-1})Q_{N_2}(q^{-1}) \tag{7.13}$$

$$\bar{C}(q^{-1}) = C(q^{-1})P_{N_2}(q^{-1}) \tag{7.14}$$

$$Q_{N_2}(q^{-1}) = 1 + q^{-1} + \ldots + q^{-N_2+1} \tag{7.15}$$

The orders of $\bar{B}$ and $\bar{C}$ are $nb + (N_2 - 1)(na + 1) - 1$ and $nc + na(N_2 - 1)$ respectively.

The multistep predictor for a given prediction range j is

$$\begin{aligned}\hat{y}^{N_2}(t+j|t) &= [1 - A_{N_2}(q^{-N_2})]\Delta_{N_2}y(t+j) + \bar{B}(q^{-1})\Delta u(t+j-1) \\ &\quad + y(t+j-N_2) + \bar{C}_j(q^{-1})(y(t) - \hat{y}(t|t-1))\end{aligned} \tag{7.16}$$

for all j between 1 and N_2.

$\bar{C}_j(q^{-1})$ consists of $\bar{C}(q^{-1})$ from $\bar{c}_j$ to the end:

$$\bar{C}_j(q^{-1}) = \bar{c}_j + \bar{c}_{j+1}q^{-1} + \ldots + \bar{c}_{nc}q^{-nc} \tag{7.17}$$

Note that the new NPC predictor is *not* equal variance. The predictions for different ranges are no longer the same. Even if the model is correct, the j-step ahead prediction error is given by

$$\epsilon_j(t) = \tilde{C}_j(q^{-1})\xi(t+j) \tag{7.18}$$

where $\tilde{C}_j(q^{-1})$ contains all the neglected terms of $\bar{C}(q^{-1})$:

$$\tilde{C}_j(q^{-1}) = \bar{c}_0 + \bar{c}_1q^{-1} + \ldots + \bar{c}_{j-1}q^{j-1} \tag{7.19}$$

The prediction error variance is therefore given by the following expression, similar to equation 7.3.

$$\sigma^2_{\epsilon_j} = \sum_{i=0}^{j-1} \bar{c}_i^2 \tag{7.20}$$

Obviously the prediction error variance is a function of prediction range once again, and so the predictor cannot be equal-varaince.

7.3.1 Control Calculation

The right hand side of equation 7.16 contains only past and present values of the plant output. This is because the first term in $1 - A_{N_2}$ is $-a_{N_2 1}q^{-N_2}$. Since $N_2 \geq j$, the latest value of y that will appear is $y(t)$. The same is obviously not true of u, so the u terms in equation 7.16 must be separated into past and future.

The first j-1 terms of $\bar{B}$ in equation 7.16 correspond exactly to the $G_j(q^{-1})$ polynomial in the GPC control action calculation. The rest of the right hand side of equation 7.16 is analogous to the f vector in GPC. This is made more obvious if equation 7.16 is rearranged thus:

$$\begin{aligned} \hat{y}^{N_2}(t+j|t) &= G_j(q^{-1})\Delta u(t+j-1) + G''(q^{-1})\Delta u(t-1) \\ &\quad + y(t+j-N_2) + [1 - A_{N_2}(q^{-N_2})]\Delta_{N_2} y(t+j) \\ &\quad + \bar{C}_j(q^{-1})(y(t) - \hat{y}(t|t-1)) \end{aligned} \tag{7.21}$$

or:

$$\hat{y}^{N_2}(t+j|t) = G_j(q^{-1})\Delta u(t+j-1) + f(t+j) \tag{7.22}$$

where $G_j(q^{-1})$ and f are defined as for GPC.

The control action calculation from this point on is exactly the same as for GPC.

7.4 Parameter Estimation for NPC

The really clever part of the NPC predictor only comes to light when process identification is examined. NPC comes from the overall control criterion in Chapter 4, so the identification objective must be the same as for LRPI:

$$J_{ID} = \sum_{j=N_1}^{N_2} (y(t+j) - \hat{y}(t+j|t))^2$$

If the one- and two-step ahead predictions are formed from equation 7.16 and compared, then an interesting fact comes to light. For the purposes of this discussion, we will set t to one, and assume that all previous process measurements have been taken. The one step ahead prediction is:

$$\begin{aligned} \hat{y}^{N_2}(2|1) &= [1 - A_{N_2}(q^{-N_2})]\Delta_{N_2} y(2) + \bar{B}(q^{-1})\Delta u(1) + y(2 - N_2) \\ &\quad + \bar{C}_1(q^{-1})(y(1) - \hat{y}(1|0)) \end{aligned} \tag{7.23}$$

and the two-step ahead prediction is given by

$$\begin{aligned} \hat{y}^{N_2}(3|1) &= [1 - A_{N_2}(q^{-N_2})]\Delta_{N_2} y(3) + \bar{B}(q^{-1})\Delta u(2) + y(3 - N_2) \\ &\quad + \bar{C}_2(q^{-1})(y(1) - \hat{y}(1|0)) \end{aligned} \tag{7.24}$$

The one step ahead prediction from 2 to 3 is given by the following:

$$\begin{aligned} \hat{y}^{N_2}(3|2) &= [1 - A_{N_2}(q^{-N_2})]\Delta_{N_2} y(3) + \bar{B}(q^{-1})\Delta u(2) + y(3 - N_2) \\ &\quad + \bar{C}_1(q^{-1})(y(1) - \hat{y}(2|1)) \end{aligned} \tag{7.25}$$

From the form of these expressions it is clear that the deterministic part of the prediction — the first line of each equation — depends only on whether $\hat{y}(2)$ or $\hat{y}(3)$ is being predicted, and not at all on whether the prediction is from time 1 or 2. In general, the first line of each equation depends only on the *absolute* time for the prediction. This is in common with the DMC predictor discussed above, and distinct from GPC. It is the major feature of NPC: the best long range predictions

are formed using the same model as the best one-step-ahead predictions. Because the same predictor is used for all prediction distances, the best predictor for one prediction distance is the same as the best for any other prediction distance. This is the most important corollary of using the equal-variance principle: not only are all predictions the same, but they can be found using nice, simple least squares. The adaptive filter of LRPI is superfluous, and because of the structure of the NPC predictor, the L filter would be equal to 1.

The remainder of the prediction is the prediction of future effects of past disturbances. It is a function of the prediction distance, as mentioned above. In practice $\bar{C}(q^{-1})$ will most likely be specified off-line in a way similar to the way the $T(q^{-1})$ filter is chosen in GPC applications. Unfortunately, the optimal $\bar{C}$ depends on P_{N_2} and therefore on $\hat{A}(q^{-1})$ (equation 7.14). Specifying it off-line will therefore take some care.

The catch when using NPC is that an enormous number of parameters need be identified. For a first order process model and $N_2 = 10$, then $\bar{B}(q^{-1})$ has 19 elements ($nb + (N_2 - 1)(na + 1)$) and $\hat{A}_{N_2}$ has one, making a total of 20 parameters to be identified. If the model were second order, then an additional 10 parameters would need to be identified. If the process had complex dynamics then the number of parameters could easily exceed 40.

Two different methods have been developed to reduce the number of parameters to be identified. They both share a common failing: no reduced order predictor will be a true equal-variance predictor. Of course, as ever, theoretical optimality is one thing and practical considerations are quite another. One method is a straightforward *ad hoc* reduction in the number of $\bar{B}$ parameters in the model. Instead of identifying and using 50 $\bar{b}$ parameters, for example, only the first N_2 parameters are identified. The remaining parameters are assumed to be zero.

An alternative method is to reduce the number of parameters to be identified through a semi-adaptive approach. The $\bar{b}$ parameters are first identified off-line, and

then only $\hat{A}_{N_2}$ and $\hat{b}_0$ are estimated on-line, where $\hat{b}_0$ is a factor by which all the fixed $\hat{b}$ parameters are multiplied: a gain.

7.5 Simulations

Lu and Fisher (1990b) describe simulations carried out to evaluate the properties of NPC in comparison to GPC. Rohrs' third order plant (Rohrs *et al.*, 1984) was used and the sample time was 0.1 seconds. For all simulations a first order model was identified, and the controller tuning parameters were set to $N_1 = 1$, $N_2 = 10$, $NU = 1$ and $\lambda = 0$.

Lu and Fisher state that the adaptive GPC controller was unable to provide stable control with the reduced order model. The GPC controller used had no T filter. It is well known that GPC cannot operate under conditions of severe model-plant mismatch without a T filter. NPC can, but NPC uses a different noise model structure than GPC and, as ever, controller performance is a function of the noise model (the denominator of the noise term is Δ_{N_2} for NPC and Δ for GPC). If the default T filter is chosen for GPC, $T(q^{-1}) = 1 - 0.8q^{-1}$, and LRPI is used as the process identification method, then the comparison is much fairer.

The closed loop performance of adaptive GPC (and LRPI) is shown compared to adaptive NPC in Figures 7.1 and 7.2. In the first figure there is no noise, but in Figure 7.2 there is a measurement noise (standard deviation = 0.1) added to the plant output. The values shown are the noise-free plant output. GPC cannot control as well as NPC, but control is still stable. Obviously the overparameterization of NPC permits better prediction in the face of model-plant mismatch, but similar control quality can be obtained using a similarly overparameterized adaptive GPC, even without a T filter.

In general, the overparameterization of the model used in NPC provides good long range prediction in the presence of significant model-plant mismatch and mea-

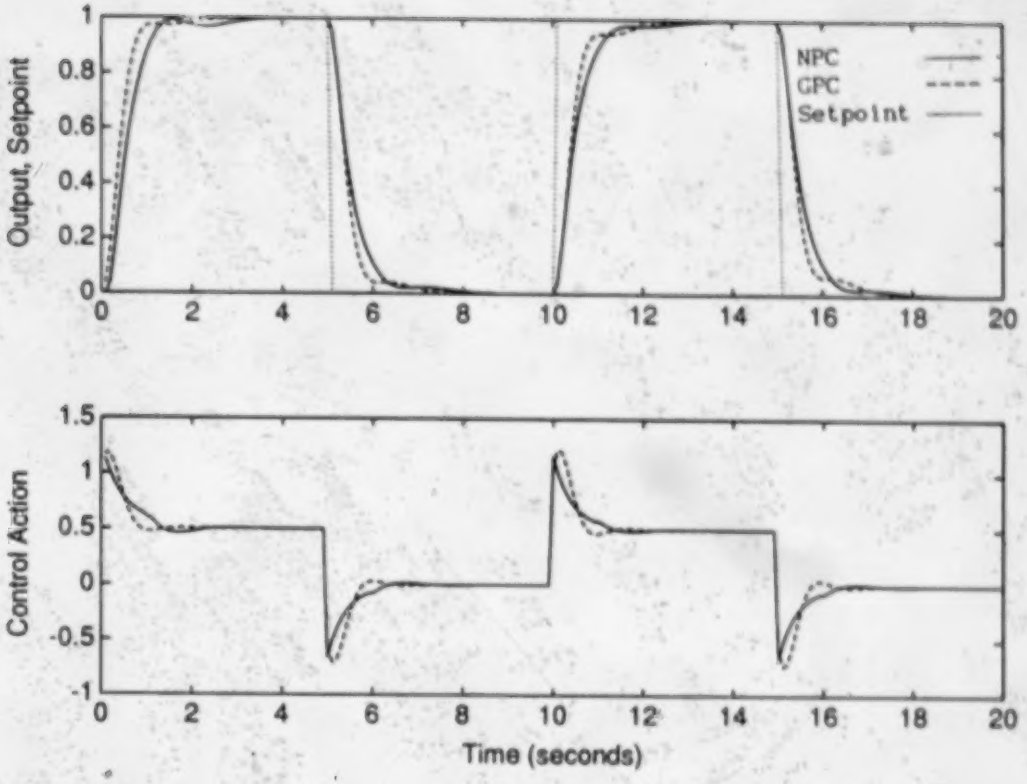

Figure 7.1: Comparison of GPC and NPC: Noise Free Case

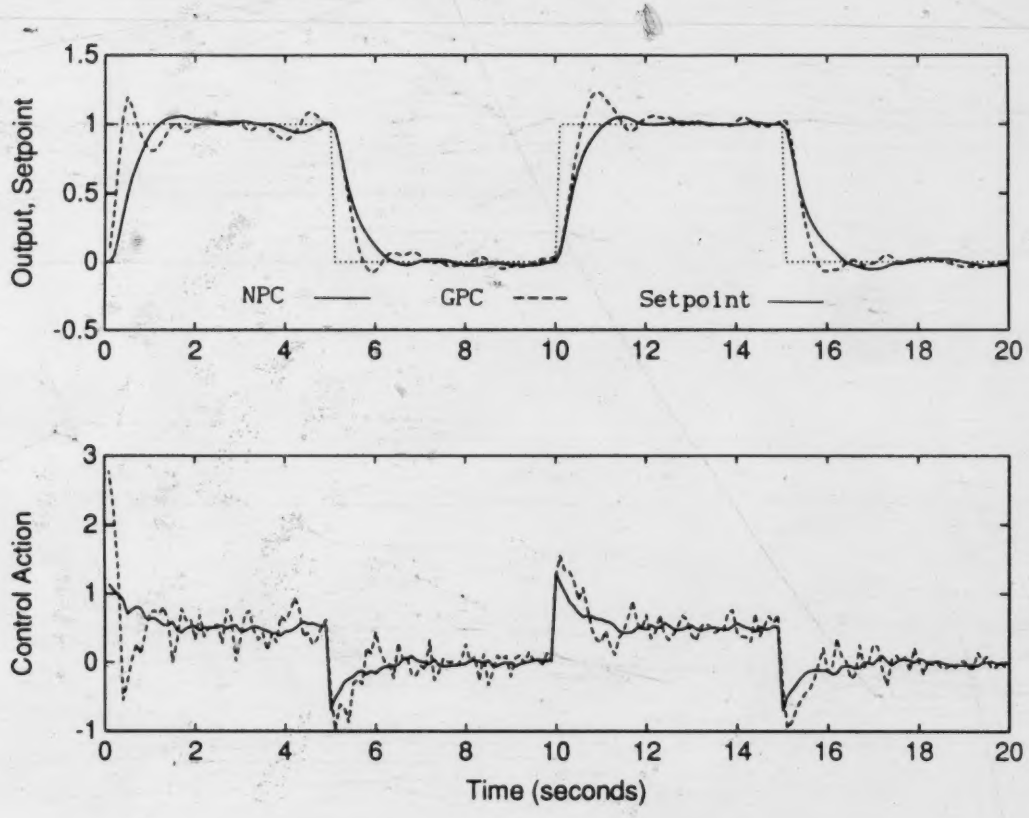

Figure 7.2: Comparison of GPC and NPC: Measurement Noise Present

surement noise. Good prediction in this case gives good control. The major stumbling block in the implementation of NPC is the computational load resulting from the large number of parameters to be identified. While NPC did perform better than GPC with a first order model, there was little difference between NPC and GPC when GPC used a similarly overparameterized model.

7.6 Conclusions

The equal-variance principle is an interesting theoretical tool for design and evaluation of predictors. Unfortunately, equal-variance is virtually impossible to obtain when a C polynomial is used.

NPC is similar to an implicit version of GPC. Its advantages and disadvantages are on the whole similar to those of implicit schemes. NPC is capable of prediction (and control) even better than GPC with LRPI, but at the expense of a greater computational load. It represents another valid approach to the problem of identification for long range prediction in adaptive control. Unfortunately, the implementation of NPC is complicated and the numerical complexity is great. The reduced-order and semi-adaptive formulations show promise, but are suboptimal and can still be computationally intensive. Pending detailed analysis NPC cannot be recommended for industrial applications.

Chapter 8

Forgetting Methods for Adaptive GPC

The question of data forgetting in adaptive control was raised during the experiments described in Chapter 3. Specifically, the concern was the effect on identification when both feedforward and feedback control were used (FF+FB or MISO control). The role of data forgetting in adaptive control was introduced in Chapter 4 and has been discussed widely in the literature (see Ljung and Gunnarsson, 1990 for a survey). When several signals are used in the model, such as for a multivariable controller, data forgetting must be examined very carefully. Whatever forgetting method is chosen, it will have to be able to cope with the fact that different parameters will vary at different rates, and different signals will contain different amounts of information.

With a MISO control configuration the controller has no influence over the feedforward variable by definition. If the feedforward variable is quiet for long periods of time, there is no way the controller can add excitation. The covariance matrix update (see Chapter 4) will have to be chosen carefully. Otherwise, the adaptation mechanism will lose confidence in the disturbance model parameters, even though there may be no good reason to do so.

In this chapter, a number of covariance matrix updating methods are exam-

ined within the framework of MISO adaptive control. Exponential forgetting methods will be shown to be less appropriate than directional forgetting methods. The discussion is restricted to RLS, although the analysis and many of the conclusions could be applied to other similar parameter estimation methods (e.g. LRPI).

8.1 Forgetting in Recursive Least Squares

The RLS algorithm makes use of the covariance matrix to store previously gathered information about the process. In fact $\mathbf{P}$ is defined as the information matrix. Data forgetting is accomplished through modifying the elements of the covariance matrix appropriately. A large value in the covariance matrix implies a large parameter variance, or a low confidence in the corresponding parameter estimate. All methods used to discard old data increase the magnitude of certain elements in the covariance matrix. Two common ways of doing so are covariance resetting and the use of forgetting factors.

Covariance resetting is performed when the present parameters are known to form an incorrect description of the process dynamics. The covariance matrix is reset to some initial condition, usually some constant multiplied by the identity matrix. In effect, the parameter estimation is re-started, albeit with a new initial guess — the existing model parameters. All information regarding the statistical interdependencies of the model parameters is discarded. There are less brutal forms of resetting, known as covariance modification, where $a\mathbf{I}$ ($a << 1$) is added to the covariance matrix, thus modifying statistical information rather than destroying it. Resetting may be appropriate if the model is badly in error, but it is usually a case of throwing the baby out with the bath water: just because the model is incorrect, it does not necessarily follow that all of the information previously gathered is useless.

Covariance resetting has been used successfully, and is worthwhile especially when the plant changes in a discrete manner: parameters suddenly jump from one

value to another after being constant for a long time. If the parameters change more or less continually, resetting the covariance matrix is a rather drastic act. Since statistical information is thrown away, the parameter estimation scheme may have to redo a great deal of work for a small benefit: the true parameters may have drifted only slightly from the old values.

An alternative to covariance resetting is the use of a forgetting factor. The major assumption in adaptive control is that the process changes with time. This being the case, it is obvious that the most recent data describe the new process, but older data may describe an older condition. Older data are therefore not as relevant as newer data. A way to utilize this knowledge is to perform the parameter estimation with more accent on the more recent data. This is accomplished through the use of "forgetting factors" which determine the rate at which old data are to be discarded.

There is still some open discussion regarding the most appropriate forgetting technique for any given implementation. The contribution of this chapter is to discuss the effects of different forgetting schemes relevant for adaptive MISO GPC. Different forgetting schemes will be analyzed from a graphical framework used by Kulhavý (1987) and Rogers (1989).

The fundamental problem of forgetting is this: data must be discarded fast enough for the model to track the process, but not too fast. If forgetting is too slow, then the adaptive controller cannot track changes in the process. If too fast, then the parameters change too quickly, in response to noise rather than actual physical changes in the process. In the noise-free case, it is possible for the covariance matrix to grow without bound. This phenomenon, known as covariance blowup, is extremely dangerous and well-described in the literature (Wittenmark and Åström, 1984). It causes "bursting," a situation where there is a sudden "burst" of control activity brought on by a combination of large covariance matrix and a prediction error (often very small). Needless to say, discarding data too quickly is the more dangerous case.

Most of the forgetting methods are formulated to prevent unbounded growth of the covariance matrix.

There is a subgroup of forgetting algorithms which aims to keep some measure of the covariance matrix constant, such as the constant trace algorithm of Sripada and Fisher (1987). LMS may be interpreted as falling into this group, since it maintains the entire covariance matrix constant, as was pointed out in Chapter 4. These approaches aim to keep the covariance matrix bounded and thus keep the adaptive gain finite. They do so, but since the trace of the covariance matrix is fixed (the entire matrix in LMS), identification cannot accelerate when necessary. These methods are therefore very conservative solutions to the forgetting problem. They cannot increase the adaptive gain at all, although they completely avoid the problem of covariance blowup.

There are two main implementation-specific questions that must be answered by the user before using any forgetting scheme. They are: (1) how noisy are the measurements, and (2) in what manner can the process be expected to change? The noise level determines the best achievable prediction quality, and the estimator must have an estimate of the noise level to tell good predictions from bad. Otherwise, it is impossible to discern "reasonable" prediction (where prediction error is caused mostly by noise) from "poor" prediction (where prediction error is caused by an inaccurate model). Virtually all forgetting factor algorithms also contain a user-specified parameter which defines the amount of information retained in the covariance matrix. Often it is a direct measure of the covariance matrix, such as the trace, or the maximum permissible eigenvalue, and sometimes it is a measure of the rate at which information is to be discarded. Either way, the user specifies the maximum rate at which the process model will be permitted to change.

In the multivariable environment the problem is further complicated by the presence of different signals which may contain different amounts of information, and the associated parameters may change at different rates. As an example, consider a

stirred tank heater: although the process time constant may change every time the feed flow rate changes, the process gain may not. Problems of measurement and parameter scaling may defeat a constant trace algorithm: the $\hat{B}$ and $\hat{D}$ parameters have units (often different) that depend on the particular application. If the parameter variances are of very different magnitudes, then constant trace methods (including LMS) may cause problems.

As was mentioned in Chapter 3, it is quite possible for a measured disturbance to remain approximately constant for a long time. During the long quiet period there will be no meaningful information on the effect of the disturbance, so it is important to retain all of the information collected during periods when the disturbance is active. A directional forgetting factor, such as that of Kulhavý (1987) or Hägglund (1983) will retain the information and maintain confidence, while an exponential forgetting factor will discard information and lose confidence, as will be seen in Sections 8.3 and 8.4.

Before different forgetting methods can be compared in any meaningful way, their effects must be quantified, through the use of parameter confidence bounds.

8.2 Parameter Confidence Bounds

The matrix **P** describes the covariances of the parameter estimates at a given instant. As such, it provides information regarding not only the parameter variances, but the correlation among the parameters. More precisely, the covariance matrix describes the χ^2 distribution of the parameters in parameter space. (For a given covariance matrix and set of parameters, if the number of samples used so far is known, then a table of χ^2 values may be used to establish confidence limits on the parameters.) A number of researchers, including Kulhavý and Karny (1984) and Rogers (1989) have used this interpretation to examine different forgetting schemes.

A given diagonal element of the covariance matrix, p_{ii}, is the variance of

the single parameter estimate $\hat{\theta}_i$. Each off-diagonal element p_{ij} is the covariance between the two parameter estimates $\hat{\theta}_i$ and $\hat{\theta}_j$. Its sign and magnitude describe the interrelationship between the two parameters. The three elements p_{ii}, p_{jj} and p_{ij} may be used to find the degree of correlation between $\hat{\theta}_i$ and $\hat{\theta}_j$. If p_{ij} is equal to zero, then the two parameter estimates are statistically uncorrelated, and a change in one does not imply any kind of change in the other. If, however, p_{ij}^2 is equal to the product of p_{ii} and p_{jj} then the two parameter estimates are perfectly correlated. Perfect correlation means that a change in one parameter is always matched by a corresponding change in the other.

A high degree of correlation between two parameter estimates can be seen relatively easily by looking at the covariance matrix itself, but a more complete picture can be seen if a graphical method is used. The covariance matrix by definition describes the χ^2 distribution of the parameter estimates. Another way of stating this is that the covariance matrix may be used with a table of χ^2 values to show the 95% confidence limits on the parameter estimates. For a single parameter estimate the confidence limits may be shown as ticks on a line, as can be seen in Figure 8.1 for a parameter a_1 of value -0.8 and variance of 0.025 after 1000 samples. The 95% confidence limits are at 1.96 standard deviations from the expected value because the χ^2 distribution approaches the normal distribution as the number of measurements tends to infinity.

If there is another parameter, say b_1, the confidence limits for it may be plotted perpendicularly to those for the first, yielding Figure 8.2 if $b_1 = 0.2$ and $\sigma_{b_1}^2 = 0.01$.

The parameters may fall anywhere within the rectangular region unless the covariance between the two parameters is specified. Without any knowledge of the independence or dependence of the two parameter estimates, no conclusions can be drawn regarding their joint distibution. If the covariance, σ_{a_1,b_1}^2, is known to be zero, then the joint 95% confidence region is the solid ellipse in Figure 8.2. If, however,

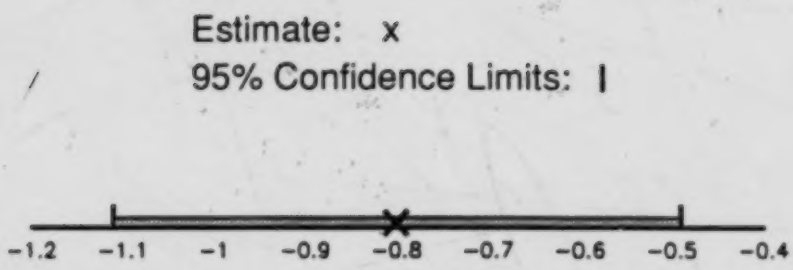

Figure 8.1: 95% Confidence Limits for a Single Parameter Estimate

$\sigma^2_{a_1,b_1} = 0.0125$, then the joint 95% confidence region is the dashed ellipse.

These ellipses serve two purposes: they define the confidence region and they also show the gain of the parameter estimation scheme by making visible the eigenvalues and eigenvectors of the covariance matrix. Briefly, the principal axes of the ellipses are in the directions of the eigenvectors of $\mathbf{P}$ and the lengths of the principal axes are proportional to the square roots of the eigenvalues of $\mathbf{P}$. In the general case, $\mathbf{P}$ may be represented by an n-dimensional hyperellipse, the principal axes of which are in the directions of the eigenvectors of $\mathbf{P}$, as before. The lengths of the principal axes of the hyperellipse are also given by the square roots of the eigenvalues. Two-dimensional projections may be used to build up an overall picture of the hyperellipse, and will be introduced in Section 8.3.1.

Each time one iteration of RLS is carried out, the parameter confidence limits change. The centre of the ellipse moves to the coordinates of the new parameter estimates and the size and shape change to reflect the updated covariance matrix. Consider the following example.

Example 8.1

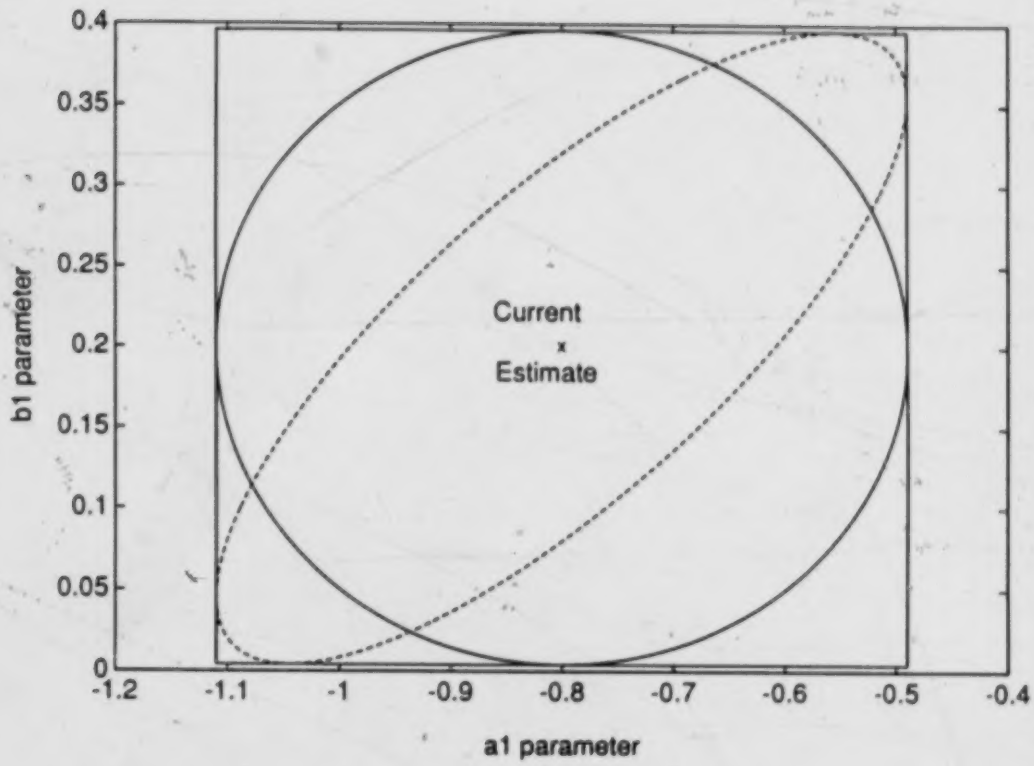

Figure 8.2: 95% Confidence Limits for Two Parameter Estimates

At time $k-1$ the model parameters and covariance matrix are:

$$\hat{\theta}(k-1) = \begin{bmatrix} -0.800 \\ 0.200 \end{bmatrix}; \quad \mathbf{P}(k-1) = \begin{bmatrix} 0.250 & 0.0125 \\ 0.0125 & 0.100 \end{bmatrix}$$

The solid ellipse and number "1" in Figure 8.3 show these initial conditions. The ellipse is shown for the 95% confidence region.

At time k the data vector and prediction are

$$\phi(k) = \begin{bmatrix} 5 \\ 4 \end{bmatrix}; \quad \hat{y}(k|k-1) = -3.20$$

If the actual plant output is -2.20, then the *a priori* prediction error is 1.00. If RLS is used with no forgetting, then the new parameter vector and covariance matrix are:

$$\hat{\theta}(k) = \begin{bmatrix} -0.661 \\ 0.250 \end{bmatrix}; \quad \mathbf{P}(k) = \begin{bmatrix} 0.0693 & -0.0518 \\ -0.0518 & 0.0771 \end{bmatrix}$$

The parameters and 95% confidence region are shown by the number "2" and the dashed ellipse in Figure 8.3. The motion of the parameters has been in the direction $\mathbf{P}(k)\phi(k)$, shown by the solid line. The covariance matrix has contracted in the direction of $\mathbf{P}(k)\phi(k)$ and has not changed at all in the perpendicular direction.

Δ

Rogers (1989) discusses the effects of several different forgetting methods on the confidence bounds. Some of the methods discussed by Rogers were constant exponential forgetting, the constant information exponential forgetting method of Ydstie *et al.* (1985), constant trace forgetting (Sripada and Fisher, 1987) and Kulhavý's (1987) directional forgetting method. Because of the work of Rogers, directional forgetting was suggested in Chapter 3 as a possible solution for the problem of directional blowup. Different forgetting techniques all affect the final covariance matrix in different ways, as will be illustrated in the next section.

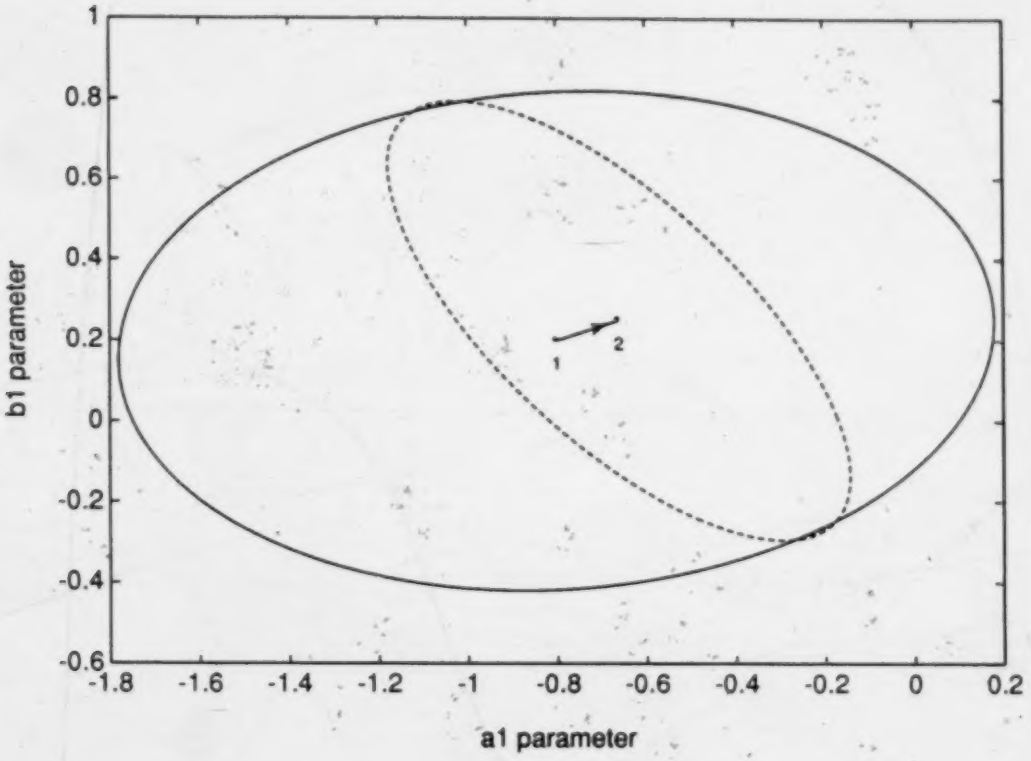

Figure 8.3: Evolution of Parameters and 95% Confidence Region for RLS

8.3 The Effects of Forgetting on Parameter Confidence

For a given plant with sampled-data inputs $u(k)$ and $v(k)$ and output $y(k)$, and an autoregressive process model with two exogenous inputs (ARXX or ARX^2), with white noise $\xi(k)$, the process model may be written as:

$$A(q^{-1})y(k) = B(q^{-1})u(k) + D(q^{-1})v(k) + \xi(k) \tag{8.1}$$

where $A(q^{-1})$, $B(q^{-1})$ and $D(q^{-1})$ are polynomials in the backshift operator, q^{-1}, of degree na, nb and nd respectively. $A(q^{-1})$ is monic and $B(q^{-1})$ and $D(q^{-1})$ have at least one leading zero element (e.g. $b_0, d_0 = 0$) from a zero order hold and/or dead time.

The standard RLS equations without forgetting are then:

$$\hat{\theta}(k) = \hat{\theta}(k-1) - \mathbf{P}(k)\phi(k)e(k) \tag{8.2}$$

$$\mathbf{P}(k)^{-1} = \mathbf{P}(k-1)^{-1} + \phi(k)\phi(k)^T \tag{8.3}$$

$$e(k) = y(k) - \phi(k)^T\hat{\theta}(k-1) \tag{8.4}$$

$$\phi(k) = [-y(k-1), \ldots, -y(k-na), u(k-1), \ldots, u(k-nb), v(k-1), \ldots, v(k-nd)]^T \tag{8.5}$$

$$\hat{\theta}(k-1) = [\hat{a}_1(k-1), \ldots, \hat{a}_{na}(k-1), \hat{b}_1, \ldots, \hat{b}_{nb}, \hat{d}_1, \ldots, \hat{d}_{nd}]^T \tag{8.6}$$

The update of the covariance matrix, P, is accomplished through the use of the matrix inversion lemma which obviates the need to invert the entire matrix and improves both numerical robustness and efficiency. Without forgetting, the covariance update is given by the following expression (from Chapter 4):

$$\mathbf{P}(k) = \mathbf{P}(k-1) - \frac{\mathbf{P}(k-1)\phi(k)\phi^T(k)\mathbf{P}(k-1)}{1 + \phi^T(k)\mathbf{P}(k-1)\phi(k)} \tag{8.7}$$

Exponential Forgetting

Forgetting methods affect the update of the covariance matrix, equations 8.3 and 8.7. The more common forgetting approach, exponential forgetting, modifies equation 8.3 to give the following relation:

$$\mathbf{P}(k)^{-1} = \lambda(k)\mathbf{P}(k-1)^{-1} + \phi(k)\phi(k)^T \tag{8.8}$$

where $0 < \lambda(k) \leq 1$. $\lambda(k)$ is the forgetting factor at time k. The actual update calculation for $\mathbf{P}(k)$ becomes:

$$\mathbf{P}(k) = \left(\mathbf{P}(k-1) - \frac{\mathbf{P}(k-1)\phi(k)\phi^T(k)\mathbf{P}(k-1)}{\lambda(k) + \phi^T(k)\mathbf{P}(k-1)\phi(k)}\right)\frac{1}{\lambda(k)} \tag{8.9}$$

The exponential forgetting methods of, for example Ydstie *et al.* (1985) and Sripada and Fisher (1987) use this update. The methods differ in the calculation of $\lambda(k)$: Ydstie's method attempts to keep constant the amount of information in the covariance matrix (the "asymptotic memory length"), while the ILS method of Sripada and Fisher keeps the trace of the covariance matrix constant.

Example 8.2

If an exponential forgetting factor, $\lambda = 0.8$, is used for the one step of Example 8.1, then the updated parameters and covariance matrix become:

$$\hat{\theta}(k) = \begin{bmatrix} -0.658 \\ 0.251 \end{bmatrix}; \quad \mathbf{P}(k) = \begin{bmatrix} 0.0816 & -0.0665 \\ -0.0665 & 0.00958 \end{bmatrix}$$

The effect of exponential forgetting on the 95% confidence limits can be seen in Figure 8.4. The confidence region is larger than for RLS with no forgetting, in all directions. The larger confidence region is a direct result of forgetting, and the fact that the increase is proportional in all directions is a result of *nondirectional* exponential forgetting.

△

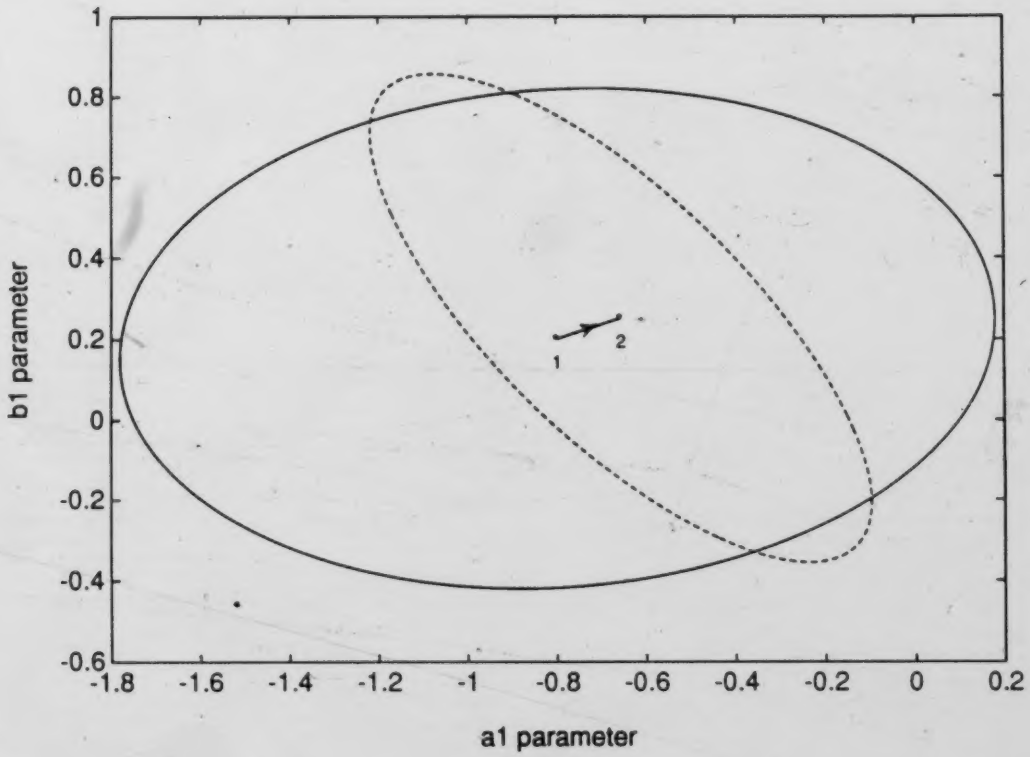

Figure 8.4: Evolution of Parameters and 95% Confidence Region for Exponential Forgetting

There are many different ways to choose the exponential forgetting factor. The method of Ydstie *et al.* (1985) attempts to keep constant the amount of information retained in the covariance matrix, as measured by the memory length. The forgetting factor, $\lambda(k)$, is calculated using the following formula:

$$\lambda(k) = \frac{n(k) + \sqrt{n^2(k) + 4w(k)}}{2} \tag{8.10}$$

where

$$w(k) = \frac{\phi^T(k)\mathbf{P}(k-1)\phi(k)}{r(k)} \tag{8.11}$$

$$n(k) = 1 - w(k) - \frac{e^2(k)}{r(k)N_0} \tag{8.12}$$

and

$r(k)$ is the estimated standard deviation of the measurement noise,
$e(k)$ is the prediction error, defined above,
N_0 is the nominal memory length of the algorithm.

There is no explicit upper bound on the covariance matrix.

One method which does impose an upper bound is the constant trace algorithm of Sripada and Fisher (1987). It is not subject to blowup, but it requires added logic to increase the gain of the estimation when the "true" or "best" parameters change.

Directional Forgetting: Kulhavý's Method

The directional forgetting methods of Kulhavý and Karny (1984) and Kulhavý (1987) use a different covariance matrix update:

$$\mathbf{P}(k)^{-1} = \mathbf{P}(k-1)^{-1} + \alpha(k)\phi(k)\phi(k)^T \tag{8.13}$$

α is the directional forgetting factor, and may take values less than or equal to 1. Zero or negative values are also permitted.

The directional forgetting factor is calculated to provide exponential forgetting in a single direction only. It is calculated from an exponential forgetting factor, $\lambda(k)$, using the following formula.

$$\alpha(k) = \lambda(k) - \frac{(1 - \lambda(k))}{G(k)} \tag{8.14}$$

and $G(k)$ is just $\phi(k)\mathbf{P}(k-1)\phi(k)$.

There are many different ways to choose $\lambda(k)$, including the method of Ydstie *et al.* above. Kulhavý and Karny (1984) leave the choice of forgetting factor to the user, but Kulhavý (1987) recommends the use of the following formula, following a Bayesian-based theoretical analysis founded on the assumption that the "true" parameters change with time.

$$\frac{1}{\lambda(k)} = 1 + (1 + \rho)\left[\ln(1 + G(k)) + \frac{G(k)}{1 + G(k)}(\hat{e}_N^2(k) - 1)\right] \tag{8.15}$$

where $\hat{e}_N^2$ is the normalized square of the prediction error:

$$\hat{e}_N^2(k) = \frac{e(k)}{\sigma^2(1 + G(k))} \tag{8.16}$$

and σ^2 is the estimated noise variance.

One interesting property of this covariance update is that $\mathbf{P}$ only grows when $\alpha(k)$ is negative, regardless of the value of $\lambda(k)$. In other words, low though the exponential forgetting factor is, as long as $\alpha(k)$ is positive the covariance matrix still contracts. This behaviour is a consequence of the form of directional forgetting: the weighting given to the new data is adjusted. The old covariance matrix is not automatically inflated as for exponential forgetting. The covariance matrix update (equation 8.13) consists of adding a positive semidefinite rank one matrix to $\mathbf{P}^{-1}(k-1)$ whenever $\alpha > 0$. This operation increases the trace of $\mathbf{P}^{-1}$. The trace of a matrix is also the sum of its eigenvalues, so the eigenvalues of $\mathbf{P}^{-1}$ are increased. The eigenvalues of $\mathbf{P}$ will therefore decrease as long as α is positive. Whenever α is negative, the matrix added to $\mathbf{P}^{-1}(k-1)$ is negative semidefinite, so $\mathbf{P}$ will grow.

It is simple to add an *ad hoc* upper bound to the covariance matrix. If the covariance matrix trace exceeds some fixed limit, then the minimum permissible value of α can be set to zero and no additional growth will take place.

The covariance update for Kulhavý's forgetting method is:

$$\mathbf{P}(k) = \mathbf{P}(k-1) - \frac{\mathbf{P}(k-1)\phi(k)\phi^T(k)\mathbf{P}(k-1)}{1/\alpha(k) + \phi^T(k)\mathbf{P}(k-1)\phi(k)} \tag{8.17}$$

Example 8.3

If a directional forgetting factor, $\alpha = 0.6$, is used for the one step of Example 8.1, then the updated parameters and covariance matrix become:

$$\hat{\theta}(k) = \begin{bmatrix} -0.584 \\ 0.277 \end{bmatrix}; \quad \mathbf{P}(k) = \begin{bmatrix} 0.0813 & -0.0475 \\ -0.0475 & 0.0786 \end{bmatrix}$$

The effect of directional forgetting on the 95% confidence limits can be seen in Figure 8.5. Once again the original values are shown by the solid ellipse and the number "1." The confidence limits from RLS with no forgetting are shown by the dotted ellipse and the corresponding parameters are at the "2." The confidence limits for directional forgetting are shown by the dashed ellipse. The new parameters from the directional forgetting update are the same as for no forgetting. The use of directional forgetting enlarges the confidence region in the direction of $\mathbf{P}(k)\phi(k)$. It does not change the size of the ellipse in the perpendicular direction. This becomes even more obvious if a negative value of α is used. The ellipse will grow to a size even larger than the initial size, but only in the direction of $\mathbf{P}(k)\phi(k)$, as can be seen in Figure 8.6.

For this case $\alpha = -0.05$. The new parameters are at the "2" and the dotted ellipse shows the new confidence region. Obviously more information has been forgotten than was added, because the confidence region is larger. The information was forgotten only in the direction of

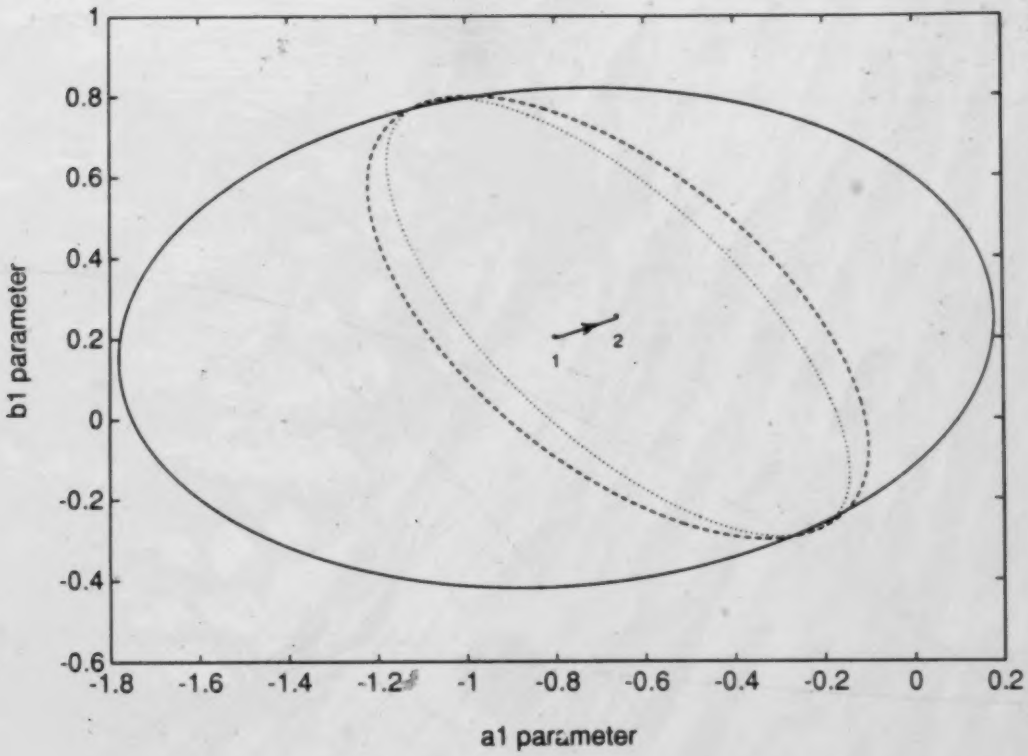

Figure 8.5: Evolution of Parameters and 95% Confidence Region for Directional Forgetting, $\alpha = 0.6$

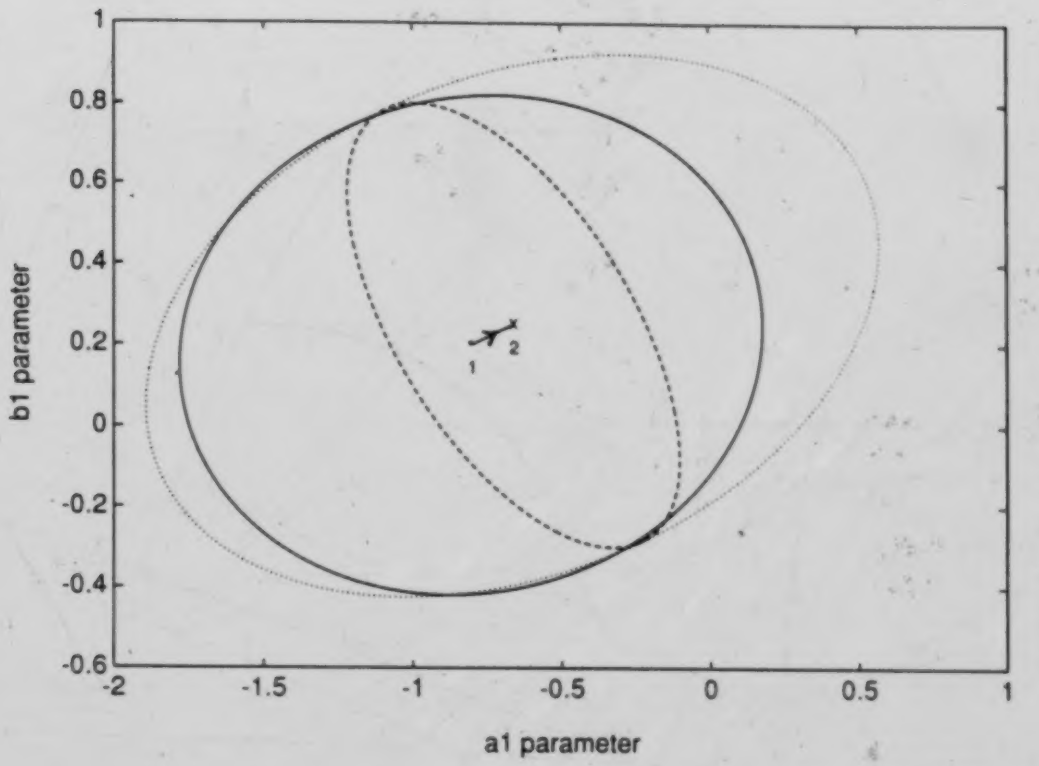

Figure 8.6: Evolution of Parameters and 95% Confidence Region for Directional Forgetting, $\alpha = -0.2$

$\mathbf{P}(k)\phi(k)$. The confidence (and therefore the adaptive gain) is the same in the direction perpendicular to $\mathbf{P}(k)\phi(k)$ as it was before the covariance update. This is the main strength of directional forgetting: if there is no information in a given direction, nothing will be forgotten in that direction.

△

Directional Forgetting: Hägglund's Method

The directional forgetting method of Hägglund(1983) uses yet another covariance matrix update:

$$\mathbf{P}(k) = \mathbf{P}(k-1) - \frac{\mathbf{P}(k-1)\phi(k)\phi^T(k)\mathbf{P}(k-1)}{1/(1/v - \alpha(k)) + \phi^T(k)\mathbf{P}(k-1)\phi(k)} \tag{8.18}$$

where $\alpha(k)$ is again the directional forgetting factor. v is a user-specified parameter, hopefully close to σ_e^2, the prediction error variance.

$\alpha(k)$ is chosen using the following algorithm:

$$\alpha(k) = \begin{cases} 0 & \alpha_d \leq 0 \\ \alpha_d & 0 < \alpha_d \leq 1/G(k) \\ 1/G(k) & 1/G(k) < \alpha_d \leq (1/v + 1/G(k)) \\ 0 & \alpha_d > (1/v + 1/G(k)) \end{cases} \tag{8.19}$$

where $G(k)$ is, as before, equal to $\phi(k)\mathbf{P}(k-1)\phi(k)$. α_d is calculated using the following pair of equations:

$$\delta_d(k) = \frac{\frac{\phi^T(k)\mathbf{P}^3(k-1)\phi(k)}{\phi^T(k)\mathbf{P}^2(k-1)\phi(k)} - a}{\phi^T(k)\mathbf{P}^2(k-1)\phi(k)} \tag{8.20}$$

$$\alpha_d(k) = \frac{1}{v} + \frac{\delta_d(k)}{\delta_d(k)G(k) - 1} \tag{8.21}$$

If the data are persistently exciting, then the covariance matrix will converge to $a\mathbf{I}$.

The objective of this forgetting method is to have the parameters converge quickly from poor initial parameters, using a large covariance matrix. As the parameters converge, the covariance matrix converges to $a\mathbf{I}$, so the identification method collapses to LMS. LMS is perfectly adequate when the parameters only drift slowly, and is easier to analyze than RLS. The disadvantage of this method is the numerical complexity of the calculation of α_d.

Like the method of Salgado *et al.*(1988) discussed next, this method was motivated by the need for better identification methods that could be theoretically analyzed. It is a heuristic method with attractive properties, rather than a method based on expected properties of the parameters. As such it is less elegant than the method of Kulhavý, but more predictable. $\mathbf{P}$ is not strictly forbidden from increasing, but it is bounded, and the bound can be calculated. Kulhavý's forgetting factor does not place an explicit bound on the magnitude of the covariance matrix, and so can theoretically lead to blowup.

Directional Forgetting: Salgado, Goodwin and Middleton

Salgado *et al.* (1988) specify the parameter and covariance updates for yet another identification approach they call the Exponential Forgetting and Resetting Algorithm (EFRA):

$$\hat{\theta}(k) = \hat{\theta}(k-1) + \frac{\alpha \mathbf{P}(k-1)\phi(k)}{1 + \phi^T(k)\mathbf{P}(k-1)\phi(k)} e(k) \tag{8.22}$$

$$\mathbf{P}(k) = \frac{1}{\lambda}\mathbf{P}(k-1) - \frac{\alpha \mathbf{P}(k-1)\phi(k)\phi^T(k)\mathbf{P}(k-1)}{1 + \phi^T(k)\mathbf{P}(k-1)\phi(k)} + \beta\mathbf{I} - \delta\mathbf{P}^2(k-1) \tag{8.23}$$

where α, β, λ and δ are user-specified constants and α is the step size or gain of the least squares algorithm. The minimum possible eigenvalue of $\mathbf{P}$ is directly related to β and the maximum possible eigenvalue is inversely related to δ. λ is a user-chosen, constant exponential forgetting factor.

As long as the eigenvalues of the covariance matrix are not at their maximum values, exponential forgetting is carried out. In addition, partial resetting takes place

in the directions perpendicular to $\mathbf{P}\phi$. In the absence of any information, $\mathbf{P}$ resets to $\bar{\nu}\mathbf{I}$ where $\bar{\nu}$ is approximately given by the following expression.

$$\bar{\nu} \approx \frac{1-\lambda}{\delta\lambda} + \frac{\beta\lambda}{1-\lambda}$$

The covariance matrix is also bounded from below. In the event that there is a great deal of persistently exciting information, the eigenvalues of $\mathbf{P}$ will never be less than $\bar{\sigma}$, given by:

$$\bar{\sigma} \approx \frac{\beta\lambda}{1+\lambda(\alpha-1)}$$

This is an interesting method, which guarantees that the covariance matrix will remain bounded, and therefore will prevent blowup. It is not however simple to apply, for two reasons. First of all, four parameters must be chosen by the user. Secondly, the covariance matrix update is no longer a rank 1 update. $\beta\mathbf{I}-\delta\mathbf{P}^2(k-1)$ must be calculated on line and then factored into UDU form before being added to $\mathbf{P}$.

8.3.1 Implications to Feedforward Adaptive Control

The preceding section has shown the effects of two different forgetting approaches on parameter confidence. The effect of increasing the size of the parameter confidence region is to increase the rate at which adaptation will take place. Exponential forgetting schemes lose confidence equally quickly in all directions, but directional schemes retain confidence in the *unexcited* direction. In this section the analysis of forgetting methods will be extended to the multivariable case, and the effects of different techniques will be demonstrated.

The dimension of the covariance matrix is $n \times n$, where n is the total number of parameters to be identified. The confidence region therefore is an ellipsoid in n-space, centred at the point $\hat{\theta}$. Two-dimensional projections will be used to show the shape of an n-dimensional ellipsoid. Consider the following example:

Example 8.4

Three parameters were to be identified, a_1, b_1 and d_1, from the process model:

$$\hat{A}(q^{-1})y(t) = \hat{B}(q^{-1})u(t-1) + \hat{D}(q^{-1})v(t-1) + \xi(t)$$

where $\hat{A}$, $\hat{B}$, and $\hat{D}$ are all first order, so the equation can be rewritten thus:

$$y(t) = [\hat{a}_1 \hat{b}_1 \hat{d}_1] \begin{bmatrix} y(t-1) \\ u(t-1) \\ v(t-1) \end{bmatrix} + \xi(t)$$

The initial parameter estimates were set to the correct values:

$$\theta^0 = \hat{\theta}(0) = \begin{bmatrix} -0.8 \\ 0.2 \\ 0.1 \end{bmatrix};$$

The initial covariance matrix was set to **I**. The initial 95% confidence region is a three-dimensional ellipsoid. It can be illustrated by displaying three two-dimensional projections, as in Figure 8.7.

The noise, $\xi(t)$, was Gaussian, zero mean and white with standard deviation of 0.1. The manipulated input, u, was a PRBS of length 1000 and the disturbance v was zero for all times so there was a great deal of information in the directions corresponding to the input and output, but none in the disturbance direction. Four different forgetting methods were examined: the exponential forgetting factors of Ydstie *et al.* and Sripada and Fisher, and the directional forgetting factors of Kulhavý and Hägglund.

When an exponential forgetting factor was used, confidence was lost in the d_1 parameter during the experiment, regardless of the method used to choose the exponential forgetting factor. When the constant information

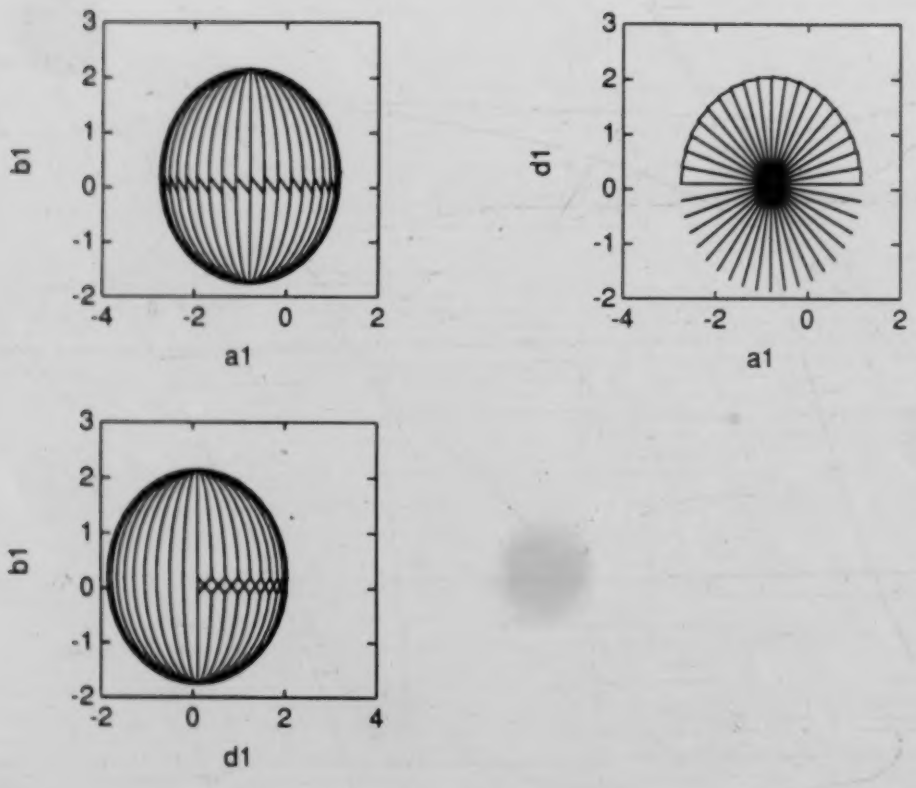

Figure 8.7: Initial Three Dimensional 95% Confidence Region

method of Ydstie *et al.* (1985) was used, the confidence region for the disturbance parameter grew without bound, as can be seen by the final confidence region, shown in Figure 8.8.

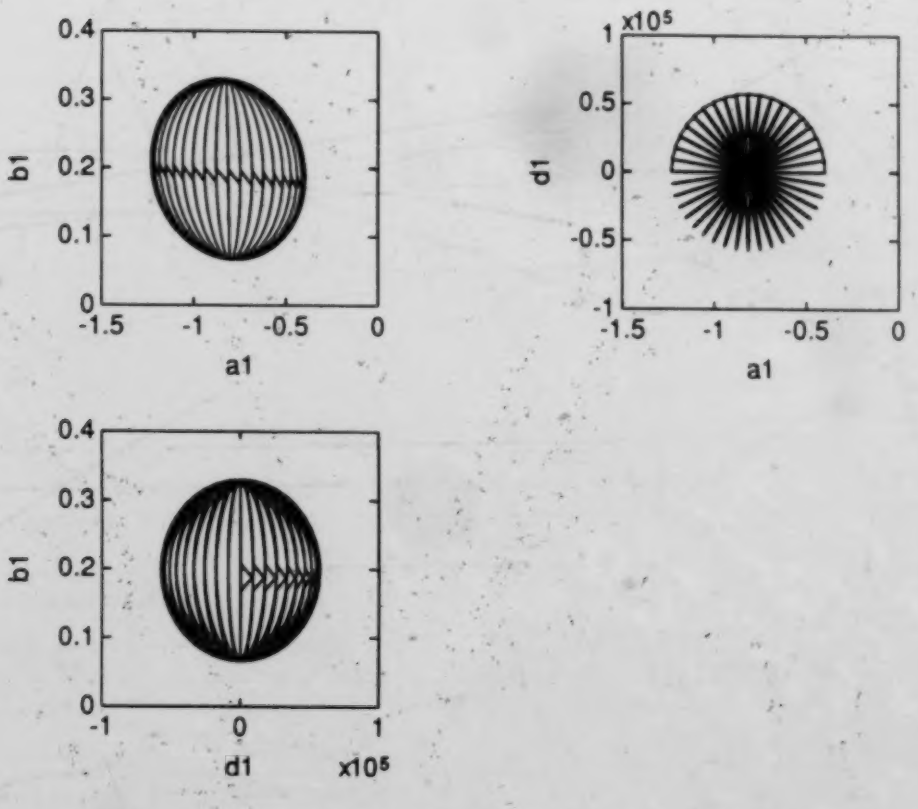

Figure 8.8: 95% Confidence Region for Ydstie's Forgetting Factor

When the constant trace method was used, then the size of the confidence region was bounded by the trace: the 95% confidence limits can never be more than $1.96 \times \sqrt{tr(\mathbf{P})}$ away from the current estimate, where $tr(\mathbf{P})$ is the trace of the covariance matrix. The adaptive gain in the d_1 direction therefore remained finite. It did, however, grow beyond the initial value, thus "blowing up," as can be seen in Figure 8.9. Moreover, the forgetting in the disturbance direction restricted the rate of adapta-

tion for the other parameters in another simulation (not shown here for reasons of brevity).

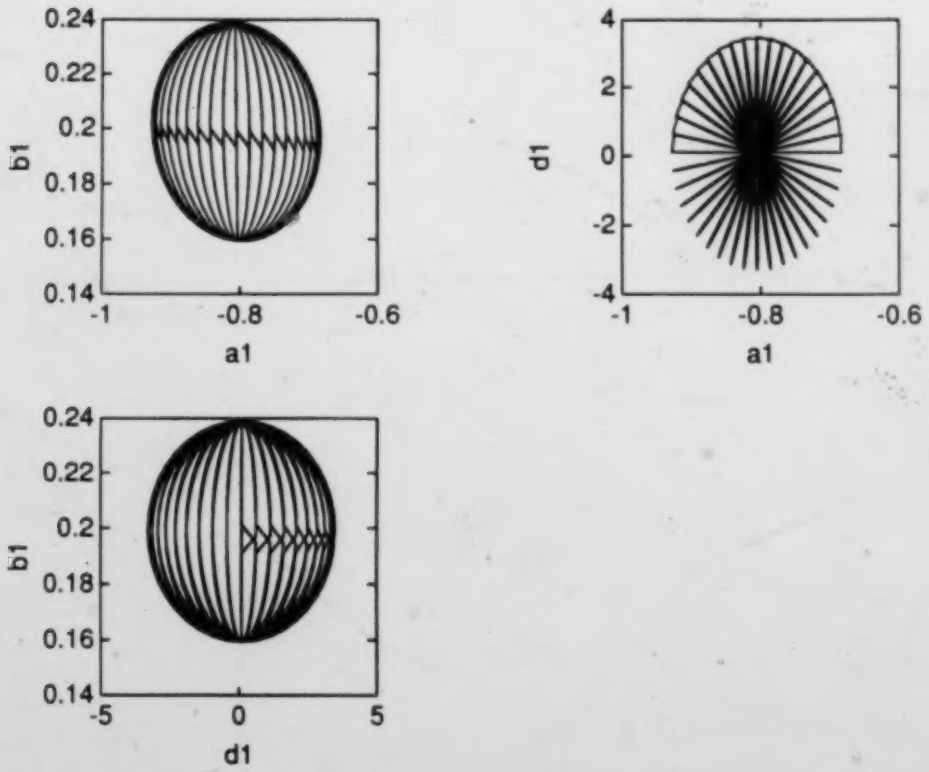

Figure 8.9: 95% Confidence Region for Constant Trace Forgetting Factor

When a directional forgetting method was used, there was no forgetting in the feedforward direction, since there was no information in that direction. Forgetting in the d_1 direction is independent of the other directions. The evolution of the confidence limits in the a_1 and b_1 directions was a function of the directional forgetting method used, but the effect on the disturbance parameter was the same: nothing can happen until there is information in that direction. This can be seen in both Figures 8.10 and 8.11.

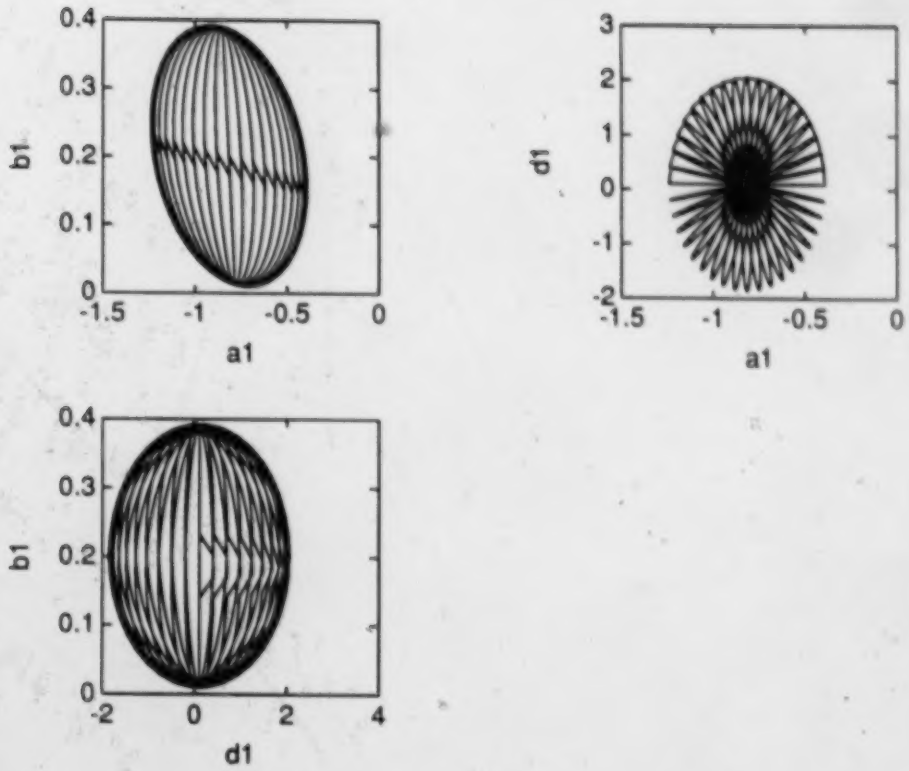

Figure 8.10: 95% Confidence Region for Kulhavý's Directional Forgetting Factor

8.4 Experimental Studies

Experimental studies were carried out on the stirred tank heater system described in Chapter 6. The main purpose of the experimental study was to determine if the use of an exponential forgetting factor could lead to directional blowup of the covariance matrix under reasonable experimental conditions and if so, whether directional forgetting would prevent directional blowup while maintaining adaptability.

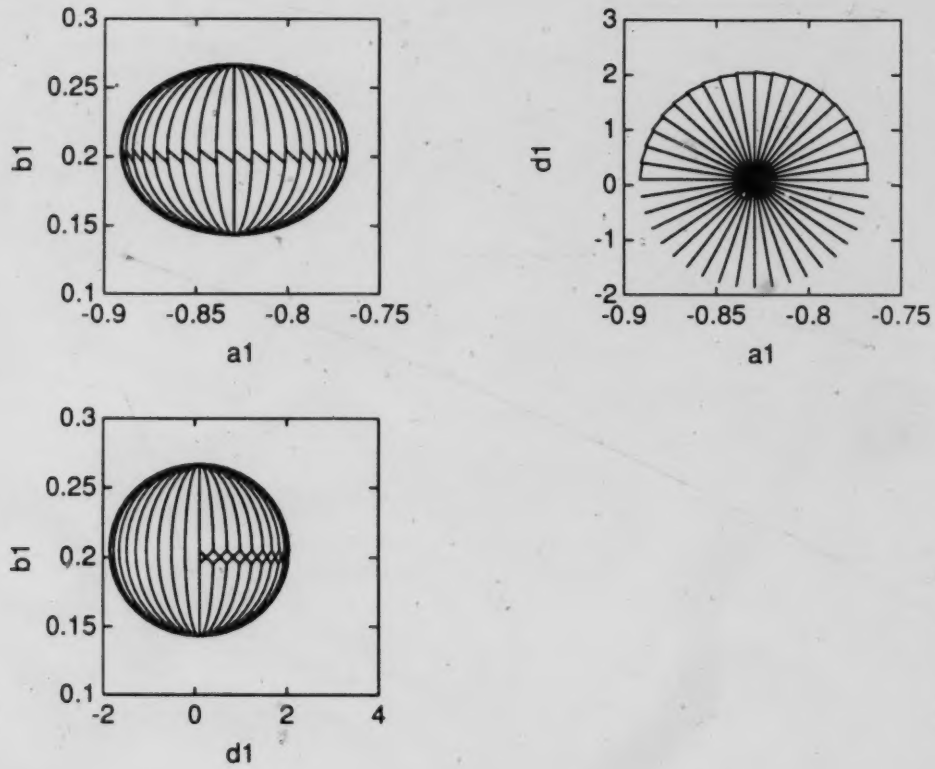

Figure 8.11: 95% Confidence Region for Hägglund's Directional Forgetting Factor

The two forgetting factors used were the constant information exponential forgetting factor of Ydstie *et al.* (1985) and the similar directional forgetting factor of Kulhavý (1987). Both are well-suited to experimental conditions, in spite of the fact that neither can guarantee that the covariance matrix will remain bounded. Both methods are easy to implement and use: the amount of programming effort required is small, and there are few user-specified parameters. They are also similar in derivation and objective (Kulhavý, 1987) and so bear comparison. Other methods were not examined experimentally because the objective of the experimental trial was to highlight the difference between exponential and directional forgetting approaches, not to provide an exhaustive examination of all forgetting methods.

The parameter Σ_0 was set to 1.0 for Ydstie's forgetting factor. Since the standard deviation of the noise was estimated to be between 0.02 and 0.05° C., this corresponds to an asymptotic memory length of from 20 to 50 samples. A prediction error deadband was also used. For all samples where the prediction error was less than 0.001° C. no update was performed. Although this seems like a preposterously small value for a prediction error deadband, it came into effect over 150 times out of the 1250 samples of the run.

Kulhavý's forgetting factor does not have an explicit prediction error deadband, but it does make use of a minimum information content deadband for numerical robustness. The information content is measured by $\phi^T \mathbf{P} \phi$. Whenever this dimensionless scalar was less than 10^{-4}, neither the covariance matrix nor the parameter estimate vector was updated. Moreover, whenever the directional forgetting factor, α, was less than 10^{-3} the covariance matrix update was considered to be pointless and was cancelled. This makes sense if equation 8.13 is examined:

$$\mathbf{P}(k)^{-1} = \mathbf{P}(k-1)^{-1} + \alpha(k)\phi(k)\phi(k)^T$$

If the absolute value of α is small enough, then $\mathbf{P}$ barely changes. The increase in precision from using the update is outweighed by the accumulation of round-off

error.

In addition to the deadbands, ρ was set to the default value of 0.2. In simulations the estimator performance has been relatively insensitive to the value of ρ. The standard deviation of the noise, σ, also needed to be specified. For the experiments described here, σ was set to 0.05° C. This was an overestimate, and was purposely conservative. σ is the primary tuning parameter of Kulhavý's forgetting method: prediction errors much larger than σ can cause extremely rapid forgetting. Of course the forgetting is always confined to a rank one subspace, rather than the entire covariance matrix, but underestimating the noise causes rapid forgetting in all excited directions.

Process inputs and outputs for the two runs are shown in Figures 8.12 and 8.13. The most obvious difference between the two runs is that the controller using directional forgetting yielded better servo response than the other configuration. This is because of the tuning of the two forgetting methods, and is really not germane to the discussion.

The setpoint and disturbance sequences were chosen to increase the likelihood of directional blowup. Following the initial tuning period the disturbance entered the process and was removed after another 50 samples. A series of setpoint changes ensured significant amounts of process excitation but no disturbance excitation for a relatively long period. The disturbance was applied again after a quiet period. Figure 8.14 shows the value of the exponential forgetting factor during the experiment. It was always quite high, and never less than 0.985, which corresponds to a memory length of over 66 samples. Comparison with Figure 8.12 shows that the low forgetting factor values coincided with the periods of great excitation, as would be expected with this forgetting method.

The directional forgetting factor behaved rather differently. The overall pattern was the same, but the details were somewhat different, as can be seen in Figure 8.15. The values of the exponential forgetting factor (which is applied in only one

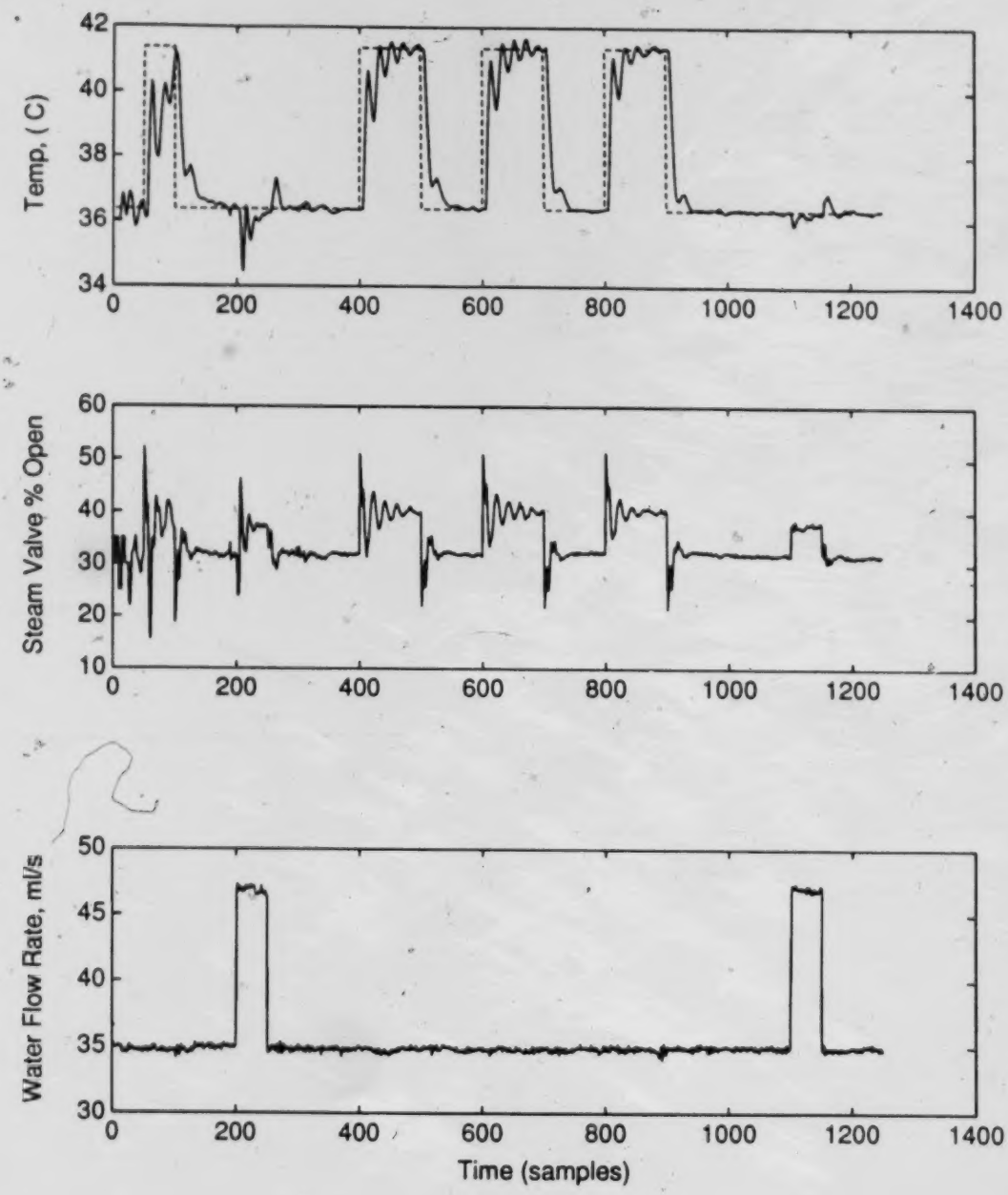

Figure 8.12: Process Inputs and Outputs for Exponential Forgetting with GPC

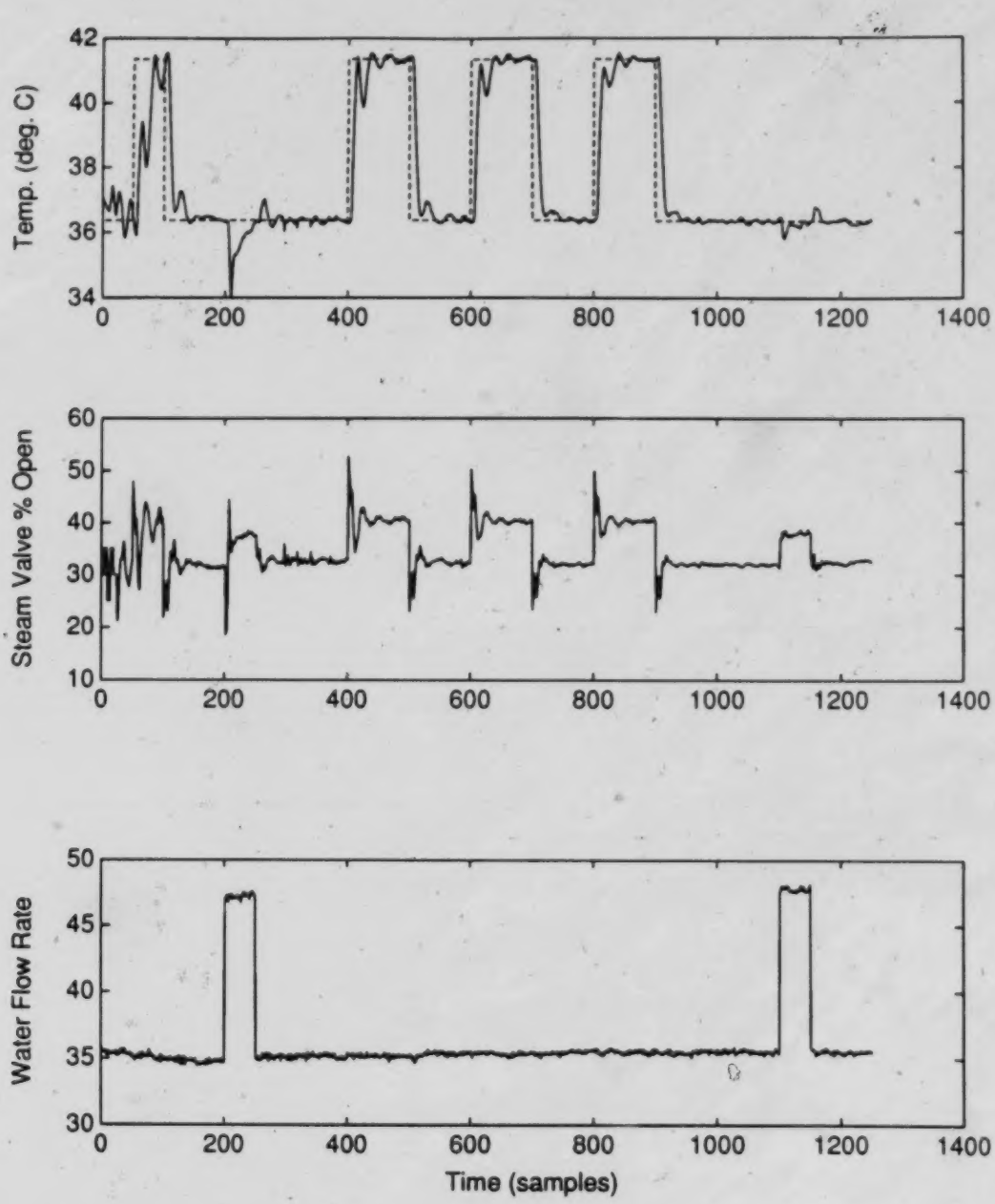

Figure 8.13: Process Inputs and Outputs for Directional Forgetting with GPC

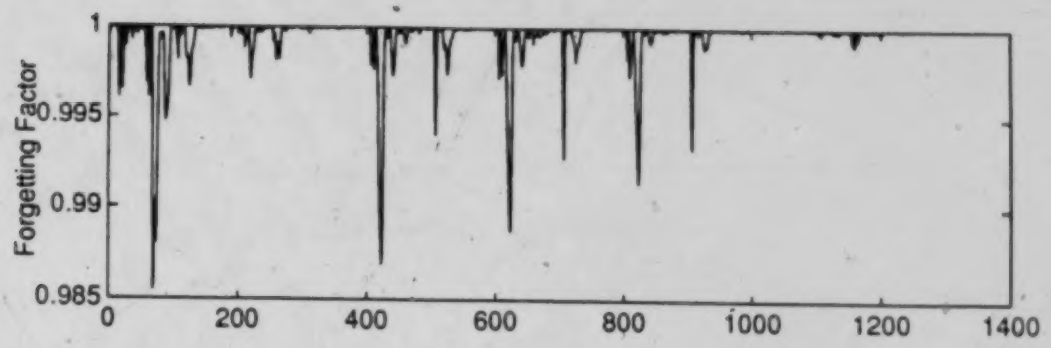

Figure 8.14: Exponential Forgetting Factor Value for Exponential Forgetting with GPC

direction) are in many cases much lower than for the first run. In particular, during the first disturbance, the forgetting factor is very low: below 0.5, corresponding to a memory length of less than two. The directional forgetting factor, α, can be seen to take on values as low as -3. It is only during the excursions of α below zero that there is actual growth in the covariance matrix, as discussed above.

Interesting though the comparison of the two forgetting approaches is, the question still remains: does the use of an exponential forgetting factor cause directional blowup? The answer is found using the confidence bounds introduced above. Figures 8.16 and 8.17 show the confidence bounds (at the 66% level, or 1.00 standard deviation) on the parameter estimates at every 50^{th} sample from 350 to 1000 samples. During this period there was little or no disturbance information, as can be seen from Figures 8.13 and 8.12.

The ellipses in the top left subplot of each figure consistently shrink with time, showing how the variances of the $\hat{a}_1$ and $\hat{b}_1$ parameters get smaller as more information is gathered over time. Both identification methods become more and more confident in these parameter estimates. In the direction of the disturbance parameters, however, the two methods differ. The directional forgetting method

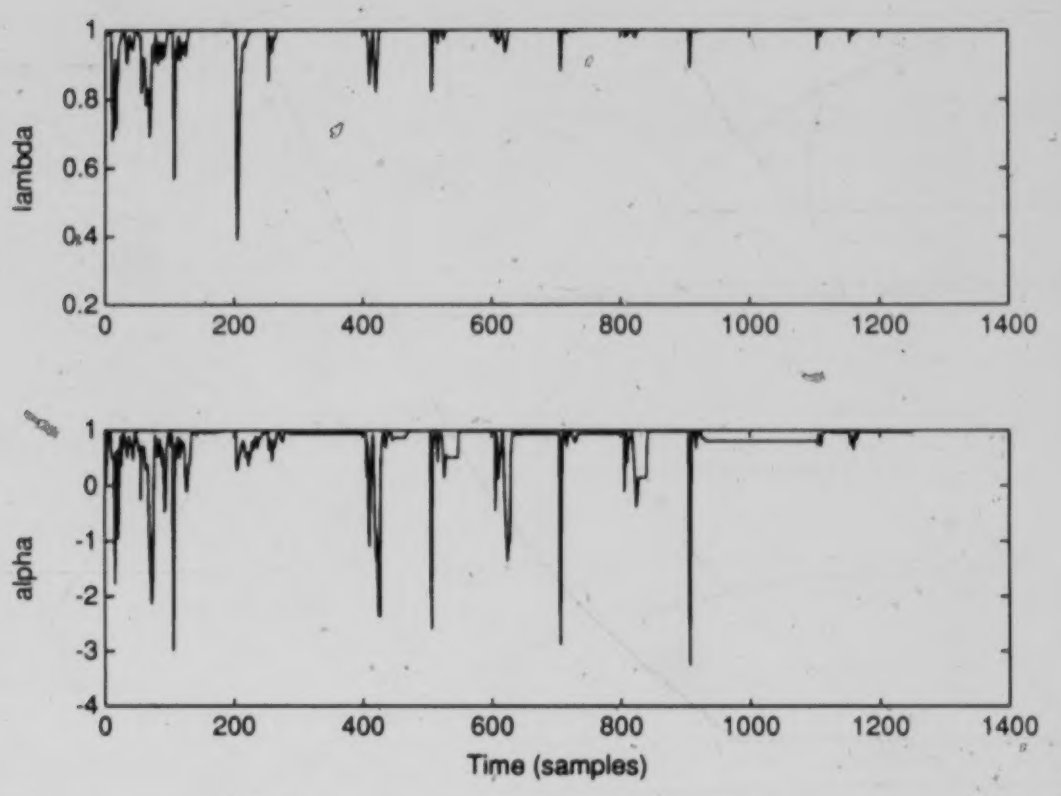

Figure 8.15: Forgetting Factor Values for Directional Forgetting with GPC

maintains confidence in the parameters while the exponential forgetting results in a slow loss of confidence.

Exponential forgetting therefore results in loss of confidence in the parameters in the absence of new disturbance information. A constant trace formulation would mitigate the loss of confidence, but would result in a loss of adaptivity in the complementary directions. Directional forgetting should therefore be used whenever there is a strong likelihood of directionally deficient information.

Another fact that is made obvious by the ellipses is the correlation between the two disturbance parameters. The nature of the correlation indicates that one of the two parameters is superfluous. The elongated shape of the $d_1 - d_2$ ellipse in both Figures 8.16 and 8.17 shows that the estimation is quite certain that the parameters lie near the line $d_1 + d_2 = -0.04$. It is however extremely uncertain where on the line the parameters lie. Because the estimation cannot discriminate betweem the two parameters, either of the d parameters could be discarded without affecting the prediction quality. A lower-order model would have provided equally good prediction and control.

8.5 Conclusions

The choice of forgetting methods for adaptive GPC implementations was examined in this chapter. There are many different forgetting methods, only a few of which were discussed here. The treatment was not intended to be exhaustive, but was intended to illustrate the advantage of directional forgetting methods over exponential methods in MISO control applications. Directional forgetting methods are preferable because of their ability to maintain parameter confidence during long periods of no disturbance acitvity.

Of the exponential forgetting methods, those that maintain constant some measure of the covariance matrix prevent unbounded directional blowup, but there

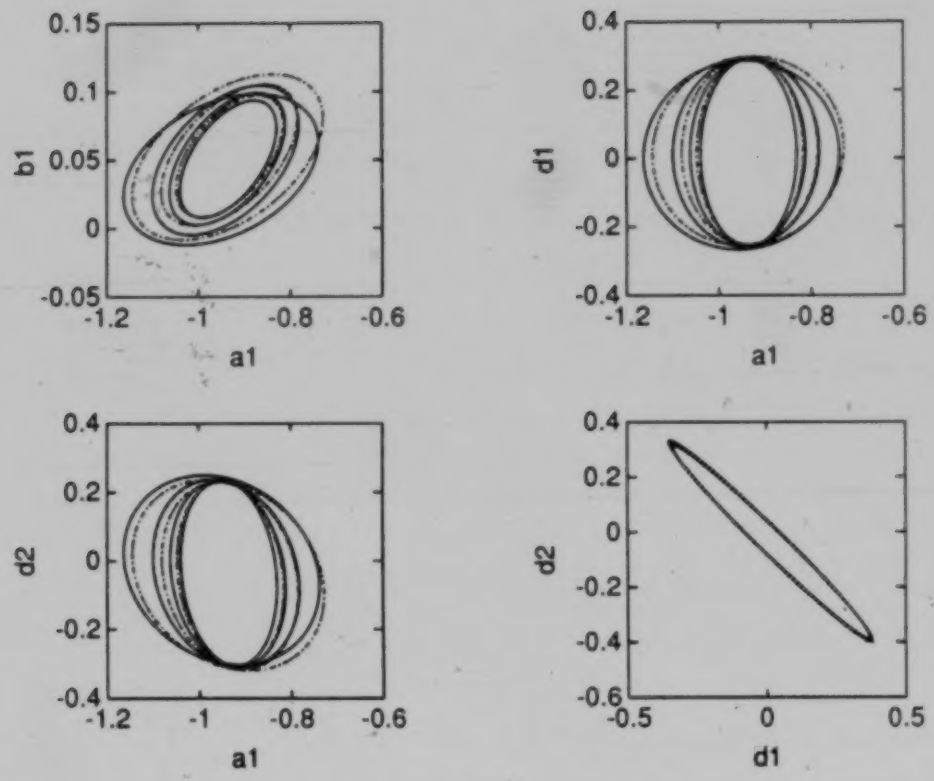

Figure 8.16: Confidence Bounds for Directional Forgetting and GPC

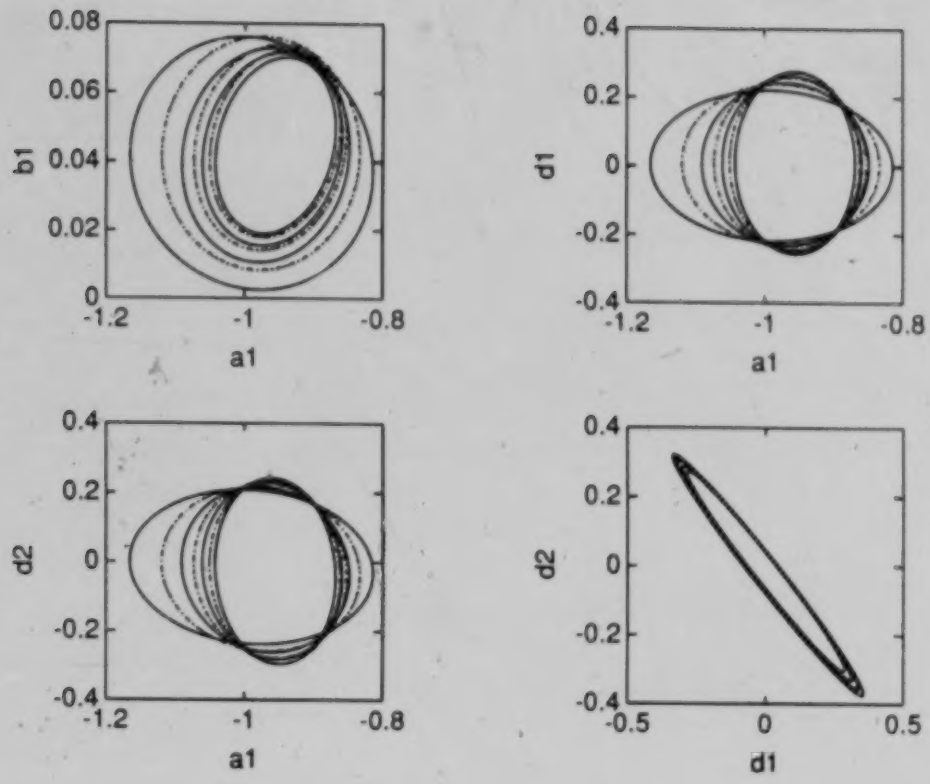

Figure 8.17: Confidence Bounds for Exponential Forgetting and GPC

is a reduction in the rate of adaptation of the other parameters. Thus the constant trace method of Sripada and Fisher is safer than the constant information method of Ydstie *et al.*, even when a prediction error deadband is used.

Hägglunds directional forgetting method is safer, but Kulhavý's method can provide faster adaptation. Hägglund's directional forgetting factor strictly constrains the value of the covariance matrix, and may be inappropriate for poorly scaled systems. There is room for personal choice between different directional forgetting methods, as well as the perennial question of performance versus safety. The method of Salgado *et al.* is a good compromise, and has in fact been used in industrial application (Allison *et al.*, 1990).

In practice within the Department of Chemical Engineering, and for research work on other facets of the adaptive controller (where forgetting is required but not the topic of the research), the directional forgetting factor of Kulhavý is recommended. Even though it does not provide an explicit bound on the adaptive gain, it provides fast adaptation, smooth parameter trajectories, and is robust in the presence of poor excitation. The only caveat for its use is that if the standard deviation of the noise is underestimated, the covariance matrix will grow too large, in an attempt to reduce the magnitudes of the prediction errors.

Chapter 9

Conclusions and Recommendations

9.1 Conclusions

This thesis has addressed a number of issues in the identification of process models for adaptive long range predictive control. The motivation for addressing process identification was that the quality of any GPC implementation is affected profoundly by the process model. Without a good process model, there cannot be good control.

The issues addressed specifically in this work were:

1. The process model identified using Least Squares, which was in theory the optimal method, could produce poor control performance. Consequently other researchers recommended the use of an additional *ad hoc* filter for the identification in spite of the lack of any theoretical support.

2. Alternative process identification methods do exist, and some may be more relevant to long range predictive control than RLS is.

3. When several signals are used in the process model, such as for MISO control, it is possible for one or more of the signals to remain quiet for an extended

period of time while others are active. In such cases, it is possible for the identification scheme to discard data too quickly in the directions of the quiet measurements. The forgetting scheme must be chosen to take this possibility into account. Of course it must also retain information in the absence of any excitation.

The contributions of the work described in this thesis are outlined below.

1. Adaptive GPC was implemented on industrial process control equipment. The performance was evaluated and it was ascertained that the controller was much easier to tune than the identification system.

2. An overall long range predictive adaptive control objective was introduced. It was used to justify the use of GPC in conjunction with an identification method (distinct from RLS) designed to minimize a control-relevant identification objective.

3. A control-relevant identification method was developed. For batch implementation, Newton-Raphson and Gauss-Newton were used to minimize the objective function.

4. Through the use of frequency-domain arguments, a feasible on-line identification method was found. The method, LRPI, uses an adaptive filter in conjunction with normal RLS. The filter depends on both the controller tuning and the model. The direct dependence of the filtering on controller tuning is result of the control-relevant identification goal.

5. The *ad hoc* data pre-filtering performed by other researchers can be explained as a suboptimal approximation to LPRI. The *ad hoc* filter often has frequency domain characteristics similar to those of the optimal LRPI adaptive filter.

6. The performance of LRPI in identifying models for adaptive GPC was confirmed through simulation studies using a standard benchmark simulation

(Rohrs' third order plant) as well as experimental studies on a pilot-scale process.

7. The Nonminimal Predictive Controller (NPC) of Lu and Fisher (1990) was compared to GPC with LRPI. NPC represents another approach to the problem of control-relevant identification for long range predictive control.

8. In adaptive feedback control, it is often possible to add excitation to improve the quality of the process model. It is impossible to add excitation in the feedforward channels, so any data forgetting method used in feedforward control is required to be more cautious than in feedback adaptive control.

9. Finally, different forgetting schemes were discussed from the point of view of combined feedforward and feedback adaptive control. Their effects on robustness in adaptive MISO control were examined and it was shown that directional forgetting schemes retain information in directions of weak or no excitation better than do exponential forgetting methods.

9.2 Recommendations for Future Work

1. As a result of the analyses carried out in previous chapters, the use of LRPI is recommended for all adaptive GPC implementations where the sample time is long enough to permit its use. In cases where high speed is crucial, RLS may be used in conjunction with a fixed data pre-filter designed using the LRPI approach and an off-line estimate of the likely process model $\hat{A}(q^{-1})$ polynomial and controller tuning. In addition, the directional forgetting factor of Kulhavý (1987) is recommended for all non-critical implementations. With the addition of an explicit limit on the covariance matrix and some means of estimating the noise level on-line, it may be used with confidence in any adaptive implementation.

2. Analysis of the LRPI filter for other model structures, especially highly oscillatory mechanical systems, is required. For this work the focus was on first order process models. First order models are completely inadequate for robot control, as the physical systems are usually only lightly damped. The LRPI filter can be band-pass or even high-pass for some underdamped process models and the effect of such filtering on the models and control must be examined more closely.

3. A more thorough comparison of GPC+LRPI with NPC is needed. In particular, the disturbance rejection properties of reduced-order NPC must be examined. The full-order version of NPC may be too numerically demanding for on-line implementation.

   The NPC noise model is significantly different from the usual Brownian motion model. The structure of the noise model will affect the disturbance rejection qualities of the controller. The actual effect of the noise model in practice should be evaluated: the different noise model may have a seriously positive or negative effect on the controller performance.

   Experimental implementation of NPC, both full-order and reduced, and comparison with GPC+LRPI, should be performed. Experimental evaluation is recommended because of the crucial role of the noise model: one primary difference in theory between GPC and NPC is the noise model, and the NPC noise model is sufficiently different from the usual ARIMAX structure to make fair simulation difficult.

4. The overall control criterion should be applied to other predictive controller methods (e.g. GPC with terminal condition weighting) to find the appropriate control-relevant identification objectives.

   The implementation of LRPI in conjunction with terminal condition weighting

presents an interesting problem: since L is theoretically infinite for steady-state prediction, it may be replaced by a denominator-type filter similar in structure to the more traditional $1/T(q^{-1})$. In such a case the filter for short range prediction would be completely different from that for steady state prediction and a multiple model approach may be more logical.

5. The effects of LRPI on the robustness of GPC to structural model-plant mismatch have not been analyzed from a robust control theory point of view. Although such analysis is most likely intractable, an approximate analysis may be possible because of the dual structure of the adaptive controller. A rigorous description of the benefits of using LRPI will also point the way to other, better, identification methods.

6. GPC may be forumlated in an implicit form, with $F_j(q^{-1})$ and $G_j(q^{-1})$ identified directly. This has not been discussed in the literature because of the vast number of parameters that would have to be identified. There is, however, a possible application of such an implicit formulation: multiple-model GPC.

   If the polynomials for each prediction distance j come from a different model (one model for each value of j), then N_p different model updates are required. Prediction (and hopefully control) would be better than when a single model is used. Of course N_p different identification problems would still have to be solved, but the problem may be tractable for the following reason. The shorter range predictions use different parameters than the long range predictions, but they use a subset of the signals used for the long range predictions. In other words, the information matrix ($\Phi^T\Phi$) for each short range prediction model is a submatrix of the information matrix for the longest range prediction model. The covariance matrix is the inverse of the information matrix, and the submatrix structure is not preserved in the inversion. The structure is preserved, however, if the matrix is stored in Cholesky factored form. Thus, although N_p

different parameter vectors must be updated, only one covariance matrix need be updated.

This implicit, multiple model formulation would have the advantage of improved prediction (and hopefully control), at the cost of some increase in numerical complexity. At worst, it would provide a benchmark against which to compare single-model GPC, and at best it could provide much better control with relatively low order models.

References

Allison, B. J., G. A. Dumont, L. H. Novak, and W. J. Cheetham (1990). Adaptive-predictive control of Kamyr digester chip level. *AIChE Journal*, **36**(7), 1075–1086.

Andersson, P. (1985). Adaptive forgetting in recursive identification through multiple models. *Int. J. Control*, **42**(5), 1175–1193.

Åström, K. J. and B. Wittenmark (1973). On self tuning regulators. *Automatica*, **9**, 185–199.

Åström, K. J. and B. Wittenmark (1989). *Adaptive Control.* Addison-Wesley, Reading, Mass.

Bierman, G. J. (1977). *Factorization Methods for Discrete-time Identification.* Academic Press, New York.

Bitmead, R. R., M. R. Gevers and V. Wertz (1990). *Adaptive Optimal Control: the thinking Man's GPC.* Snodfart Press.

Bohm, J., A. Halouskova, M. Kárný, and V. Peterka (1984). Simple LQ self-tuning regulators. Proc. 9th IFAC World Congress, 171–176, Budapest.

Clarke, D. W. and P. J. Gawthrop (1979). Self-tuning control. *IEE Proc. D*, **126**(6), 633–640.

Clarke, D. W. and C. Mohtadi (1989). Properties of Generalized Predictive Control. *Automatica*, **25**(6), 859–875.

Clarke, D. W., C. Mohtadi, and P. S. Tuffs (1987a). Generalized predictive control — part i. the basic algorithm. *Automatica*, **23**(2), 137–148.

Clarke, D. W., C. Mohtadi, and P. S. Tuffs (1987b). Generalized predictive control — part ii. extensions and interpretations. *Automatica*, **23**(2), 149–160.

Feldbaum, A. A. (1965). *Optimal Control Theory*. Academic Press, New York.

Francis, B. A. and W. M. Wonham (1976). The internal model principle of control theory. *Automatica*, **12**(5), 457–467.

Goodwin, G. C. and K. S. Sin (1984). *Adaptive Filtering, Prediction and Control*. Prentice-Hall, Englewood Cliffs, N.J.

Hägglund, T. (1983). The problem of forgetting old data in recursive identification. *Proceedings of IFAC Workshop on Adaptive Systems in Control and Signal Processing*, 213–214, San Francisco.

Ishitobi, M., S. L. Shah and D. G. Fisher (1989). Exponentially stable model reference adaptive controller with deterministic disturbances. *Proceedings of the American Control Conference*, 1852–1856, Pittsburgh.

Jacobs, O. L. R. and J. W. Patchell (1972). Caution and probing in stochastic control. *International J. Control*, **16**(1), 189–199.

Kalman, R. E. (1960). On the general theory of control systems. *Proc. First IFAC Congress, Moscow*, 481–492, London, Butterworths.

Kulhavý, R. (1987). Restricted exponential forgetting in real-time identification. *Automatica*, **23**(5), 589–600.

Kulhavý, R. and M. Kárný (1984). Tracking of slowly varying parameters by directional forgetting. *Proc. 9th IFAC World Congress*, 687–692, Budapest.

Kwok, K. (1990). Personal communication.

Lambert, E. P. (1987) *Process Control Applications of Long Range Prediction*. D.Phil. thesis, University of Oxford.

Lambert, M. (1987) *Adaptive control of flexible systems.* D.Phil. thesis, University of Oxford.

Lau, E. K. C. (1990). *Predictive control of processes with time delay.* M.Sc. thesis, University of Alberta.

Ljung, L. (1987). *System Identification: Theory for the User.* Prentice-Hall, Englewood Cliffs, N.J.

Ljung, L. and S. Gunnarsson (1990). Adaptation and Tracking in System Identification — A Survey. *Automatica*, **26**,(1), 7–21.

Ljung, L. and T. Söderström (1983). Theory and Practice of Recursive Identification. MIT Press, Cambridge, Mass.

Lu, W. (1989). *Parameter Estimation and Multirate Adaptive Control.* Ph.D. thesis, University of Alberta.

Lu, W. and D. G. Fisher (1990a). Nonminimal model based long range predictive control. *Proceedings of the American Control Conference*, 1607–1613, San Diego, CA.

Lu, W. and D. G. Fisher (1990b). Nonminimal predictive control. submitted to *Chem. Eng. Sci.*

Lu, W., D. G. Fisher, S. L. Shah and C. Mohtadi (1990). Nonminimal model based output predictors. *Proceedings of the American Control Conference*, pages 998–1003, San Diego, CA.

McIntosh, A. R. (1988). Performance and tuning of adaptive generalized predictive control. M.Sc. thesis, University of Alberta.

McIntosh, A. R., S. L. Shah, and D. G. Fisher (1990) Experimental evaluation of adaptive control in the presence of disturbances and model-plant mismatch.

S. L. Shah and G. Dumont, editors, *Adaptive Control Strategies for Industrial Use*, volume 137 of *Springer-Verlag Lecture Notes in Control and Information Sciences*, 145–174. Springer-Verlag, New York.

Mohtadi, C. (1986). *Advanced Self-Tuning Algorithms.* D.Phil thesis, University of Oxford.

Mohtadi, C. (1989). Personal communication.

Mohtadi, C. (1990) On the role of prefiltering in parameter estimation and control. S. L. Shah and G. Dumont, editors, *Adaptive Control Strategies for Industrial Use*, volume 137 of *Springer-Verlag Lecture Notes in Control and Information Sciences*, 121–144. Springer-Verlag, New York.

Mosca, E. and G. Zappa (1986). Convergence of multipredictor STR under mismatching conditions. *Proceedings of 25th IEEE Conference on Decision and Control*, 1542–1545, Athens.

M'Saad, M., M. Duque, and E. Irving (1987). Thermal process robust adaptive control: an experimental evaluation. *Proc. 10th IFAC World Congress*, Munich.

Ortega, R. and Y. Wang (1989). Robustness of adaptive controllers — a survey. *Automatica*, **25**(5), 651–677.

Panuska, V. (1968). A stochastic approximation method for identification of linear systems using adaptive filtering. *Preprints of the JACC*, 1014–1021. The University of Michigan.

Patchell, J. W. and O. L. R. Jacobs (1971). Separability, neutrality and certainty equivalence. *International J. of Control*, **13**(2), 337–342.

Priestley, M. B. (1981). *Spectral Analysis and Time Series.* Academic Press, London.

Rogers, M. D. (1988). Models and methods for recursive identification. M.Sc. thesis, University of Alberta.

Rohrs, C. E., M. Athans, L. Valavani, and G. Stein (1984). Some design guidelines of discrete-time adaptive controllers. *Automatica*, **20**(5), 653–660.

Salgado, M. E., G. C. Goodwin and R. H. Middleton (1988). Modified least squares algorithm incorporating exponential resetting and forgetting. *International Journal of Control*, **47**(2), 477–491.

Shah, S. L. and G. Dumont, editors (1990). *Adaptive Control Strategies for Industrial Use*, volume 137 of *Springer-Verlag Lecture Notes in Control and Information Sciences*. Springer-Verlag, New York.

Shook, D. S., C. Mohtadi, and S. L. Shah (1989). Identification for long range predictive control. *IEE Proc. D.* accepted for publication.

Shook, D. S., C. Mohtadi, and S. L. Shah (1990). Adaptive filtering and GPC. *Proceedings of the American Control Conference*, pages 556–561, San Diego, CA.

Shook, D. S. and S. L. Shah (1989). Adaptive GPC user's guide. Technical report, Dept. of Chem. Eng., University of Alberta.

Sripada, N. R. and D. G. Fisher (1987). Improved least squares identification. *Int. J. Control*, **46**, 1889–1913.

Stanley, W. D., G. R. Dougherty, and R. Dougherty (1984). *Digital Signal Processing*. Reston Publishing Co. (Prentice-Hall), Reston, Va., 2 edition.

Tse, E. and M. Athans (1972). Adaptive stochastic control for a class of linear systems. *IEEE Trans. Auto. Cont.*, **AC-17**(1), 38–51.

Tuffs, P. S. (1985). *Self-tuning control: algorithms and applications.* D.Phil. thesis, University of Oxford.

Wahlberg, B. and L. Ljung (1986). Design variables for bias distribution in transfer function estimation. *IEEE Trans. Auto. Cont*, **AC-31**(2), 134–144.

Widrow, B. and S. Stearns (1985). *Adaptive Signal Processing.* Prentice-Hall, Englewood Cliffs, New Jersey.

Wieslander, J. and B. Wittenmark (1971). An approach to adaptive control using real-time identification. *Automatica*, **7**(2), 211–217.

Wittenmark, B. (1975). Stochastic adaptive control methods: a survey. *International J. of Control*, **21**(5), 705–730.

Wittenmark, B. (1990). Adaptive control: Implementation and application issues. S. L. Shah and G. Dumont, editors, *Adaptive Control Strategies for Industrial Use*, volume 137 of *Springer-Verlag Lecture Notes in Control and Information Sciences*, 103–120. Springer-Verlag, New York.

Wittenmark, B. and K. J. Åström(1984). Practical issues in the implementation of self-tuning control. *Automatica*, **20**, 595–605.

Ydstie, B. E., L. S. Kershenbaum, and R. W. H. Sargent (1985). Theory and application of an extended horizon self-tuning controller. *AIChE Journal*, **31**(11), 1771–1780.

Appendix A

Lrpiest User's Guide

A.1 User's Overview

A.1.1 Scope of This Manual

Lrpiest is a program designed to operate within Multicon, a University of Alberta Department of Chemical Engineering software package designed to facilitate advanced control design and implementation. Multicon provides process and operator communications as well as scheduling and watchdog timing. Multicon operates under the QNX operating system. This document is not intended to provide an introduction to either Multicon or QNX; it is assumed that the reader is reasonably familiar with the operation of both. An introduction to both may be found in Lau (1990).

Lrpiest provides the user with an on-line process identification package particularly suited to adaptive long range predictive control. In particular, it was written with adaptive generalized predictive control (GPC) in mind, and so will be easiest to use with adaptive GPC. The LRPI technique evolved because of a need to provide better models for GPC, and so it is most obviously associated with GPC. It will still be useful for other adaptive controllers, but the notation used is consistent with the GPC literature.

A.1.2 Algorithms and References

The basic parameter estimation technique in Lrpiest is Recursive Least Squares, or RLS. The user is directed elsewhere for a description of RLS and its evolution, e.g. Ljung (1987). The particular variant applied is the UDU' factorization of Bierman (1977), and the actual covariance and parameter update routine is stolen from Ljung and Sõderstrõm(1983). There is the choice of two forgetting factors: the constant information exponential forgetting factor of Ydstie *et al.* (1985) or the directional forgetting factor of Kulhavý (1987). Filtering is provided by an *ad hoc* filter chosen by the user (cf. McIntosh, 1988) or by LRPI.

A.1.3 Use of the program

In its current form, 48 different mnemonics are required in the system data table for Lrpiest. The need for many of them can be removed by an experienced programmer, as discussed in Section A.2. The required mnemonics are shown in Table A.1.

The process model should be self-explanatory, except that c0 to c3 are used only if an auxiliary variable is being measured for feedforward control, and if feedforward is not being used, nc and f_D1 should be set to zero. (f_D1 is the feedforward control flag).

STtc, STsc and STfl correspond to the stirred tank temperature (controlled variable), steam valve setting (manipulated variable) and inlet water flow (measurable disturbance), respectively.

The mnemonics n1 and n2 are the GPC prediction horizons N_1 and N_2. They are only required when LRPI filtering is used. The maximum value of n2 is 20 or delb + 10 (whichever is smaller). t1 and t2 are the coefficients in the GPC controller T filter, which is assumed to be monic.

Iest controls the selection of initial values. If Iest=1, then they are read from the system data table, otherwise if Iest=0 the values are the programmed-in defaults.

Mnemonic	Use
na, nb, nc	process model orders
a1 – a3	process model denominator
b1 – b10	process model numerator
c0 – c3	process model feedforward term
delb, delc	process model delay
STtc, STsc, STfl	measurements
n1, n2, t1, t2	controller and LRPI tuning
lest, f_dat, f_brk, f_D1	flags
EstON	algorithm selection
Dinit	initial value of covariance matrix
Tn0–Tn2, Td0–Td2	filter constants
Sigma	identification tuning
updb, lodb	identification tuning
del1, del2, rho	identification tuning

Table A.1: System Data Table Mnemonics Required for Lrpiest

f_dat controls the recording of data. If f_dat=1 then masses of data are stored to files in the directory /user/dave/data. Setting f_brk to 1 closes the data files and turns off identification. f_D1 is set to 1 when it is desired to identify the effect of inlet flow rate on temperature.

EstON is chosen by the user, and may take one of five values:

Value	Meaning
0	Identification off
1	*ad hoc* filtering and exponential forgetting
2	*ad hoc* filtering and directional forgetting
11	LRPI and exponential forgetting
12	LRPI and directional forgetting

Table A.2: Values for EstON

Dinit is the initial value of the covariance matrix, **P**. **P**(0) is equal to Dinit times the identity matrix.

Tn0–Tn2 and Td0–Td2 are the filter constants for any *ad hoc* filtering done for the identification. The measurements are filtered by $Tn(q^{-1})/Td(q^{-1})$. No adjustments are made for steady state gain, and it is important to keep Δ as a factor of Tn if GPC is being used.

Sigma has two uses. If exponential forgetting is used, then Sigma is the Σ_0 parameter of Ydstie *et al.*(1985). Otherwise it is the standard deviation of the noise (or the best available guess). If *ad hoc* filtering is used, Sigma will have to be multiplied by the steady state gain of $Tn(q^{-1})/Td(q^{-1})$ (or at least divided by the steady state gain of $Td(q^{-1})$ if Tn is equal to Δ).

The mnemonics updb and lodb are the maximum and minimum prediction error deadbands for the exponential forgetting factor. They are not used when the directional forgetting factor is chosen. The prediction error deadbands are not particularly easy to choose, and are best found following some analysis of the prediction

errors.

The last three mnemonics, del1, del2 and rho, are the tuning parameters for the directional forgetting factor. They correspond to Kulhavý's δ_1, δ_2 and ρ, but del1 and del2 are the logarithm (base 10) of δ_1 and δ_2 respectively. The best thing to do is probably to leave them at their current values unless you are investigating the forgetting method itself.

At present, the executable file exists in the directory /user/dave/run, along with many Multicon configuration files. To use lrpiest in its present configuration (along with gpc11 and the program "setpoint") simply start Multicon, set the initial directory to /user/dave/run and choose the data file lrpi_gpc2.

A.2 Programmer's Overview

The source code for Lrpiest resides in eleven files: two "header" or .h files and nine .c files. There is a makefile in the directory /user/dave/exp which is used to keep everything up to date. If you do not already use "make" it is recommended that you do so.

The files and their purposes are outlined in Table A.3.

Of the source files only calclrpi.c contains more than one function. The functions required by all configurations are described in the next section. They include lrpiest.c, comp_est.c, form_data.c and update.c. The forgetting factor routines will then be discussed. Both dff.c and yff.c will be described, and also kg_only.c as it is only required by the directional forgetting factor. Then the filtering will be described, including calclrpi.c and pre_filter.c.

File	Purpose
calclrpi.c	Calculate LRPI Filter
comp_est.c	Compute estimation error
dff.c	Calculate directional forgetting factor
form_data.c	Form data (regressor) vector from measurements
kg_only.c	Calculate Kalman gain only for parameter update
lrpiest.c	Main routine. Does far too much
pre_filter.c	apply *ad hoc* filter to data
update.c	update covariance matrix and parameter estimates
yff.c	Calculate exponential forgetting factor
par_defs.h	Provides limits for dimensioning of arrays
r_w.h	Provides interface to Multicon system

Table A.3: Source Files for Lrpiest

A.2.1 Common Routines

lrpiest.c

This is the main routine. It deals with all communication with Multicon. All data files are opened and closed by this function. In addition, the filtering and forgetting functions are called directly. This routine was stole from an old estimation task, Parest, which provided only exponential forgetting and *ad hoc* filtering, so the details of the update were in the main routine. For Lrpiest it was necessary to add user choices for filtering and forgetting, and rather than write the whole thing from the beginning, the necessary new logic was just added in the main routine. It is ugly, but it works. As long as Lrpiest is not added to in an irresponsible manner, it should continue to work.

form_data.c

This forms the data vector phi using the filtered measurements and the model polynomial orders and delays.

comp_est.c

This five-line routine just calculates the prediction error. All input values (new measurement and regressor) must have been filtered already.

update.c

This file contains the function ud_update. ud_update calculates the new covariance matrix in UDU' factored form and the Kalman gain vector. UDU' form is used to keep P symmetric positive definite in the presence of roundoff error. If P should lose its symmetric positive definite quality then the parameter update would be impossible: the update would be a square root of a negative number, which is physically meaningless. The actual algorithm used here is taken from Ljung and Söderström (1983), p. 334. There are no user-serviceable parts enclosed. Do not modify this subroutine unless you know exactly what you are doing.

A.2.2 Forgetting Factor Implementation

Lrpiest at present gives the user choice of two forgetting factor algorithms: the constant information exponential forgetting factor of Ydstie *et al.* (1985) and the directional forgetting factor of Kulhavý (1987). The original program from which Lrpiest was developed, Parest, only had the exponential forgetting factor. The directional forgetting option was added to investigate data forgetting in MISO control.

Exponential Forgetting Factor

The exponential forgetting factor is only one part of the exponential forgetting algorithm. The prediction error deadbands play an important part in keeping the identification on track. The limits are implemented in the main routine, as part of the legacy of Parest. Large prediction errors are treated as outliers: they are caused by something that the model does not take into account, and are therefore ignored. Small prediction errors show that the parameters are close to correct. Since the forgetting factor is never quite 1.0, it is possible for covariance blowup to happen during long periods when the predictions are good and there is little or no excitation. Choice of deadband values is a tricky and *ad hoc* procedure. Good luck.

If the prediction error is within the limits, the forgetting factor is calculated and applied.

The file yff.c contains the function calc_forget_factor. Before the actual forgetting factor can be calculated, the value $G = \phi^T P \phi$ must be evaluated. Since P is stored in UDU' factored form, this takes a few steps. The forgetting factor itself is a relatively simple funciton of G and the tuning parameter Sigma.

Once the forgetting factor has been calculated, control is returned to the main program, which then constrains it to be greater than some minimum value (tpically between 0.8 and 0.95). This prevents accidental loss of information through forgetting too quickly, which tends to happen when there is a very large prediction error. The prediction error maximum helps the situation, but this acts as insurance.

Directional Forgetting Factor

The directional forgetting factor of Kulhavý (1987) is implemented in the file dff.c. Four parameters must be chosen by the user, but fortunately only one or two actually require tuning. Specifically, del1, del2, rho and Sigma must be specified. Sigma in this case is the standard deviation of the noise (measurement and/or process).

del1 and del2 are the (base 10) logarithms of the tuning parameters δ_1 and δ_2. δ_1 is a measure of the minimum information content required for the update to be performed. It functions as a deadband, but on information rather than prediction error. δ_2 is another deadband. It represents the minimum absolute value of α, the directional forgetting factor, for which the covariance matrix will be updated. If α is less than δ_2, then the parameters are updated, but the covariance matrix is not. The assumption is that more information would be lost through roundoff error than would be added by the covariance update. This is a compromise solution which works well in practice. Typical values for δ_2 are between 10^{-3} and 10^{-5}. When the parameters are to be updated and the covariance matrix is not, the standard update subroutine cannot be used, so the subroutine kg_only.c is called instead. It just calculates the Kalman gain, without performing a covariance update.

Future Forgetting Factors

Implementation of any other forgetting factors is the responsibility of the programmer. Obviously Lrpiest was written specifically for the two methods described above, and care must be taken in adding any other forgetting factors. If any are added, it is recommended that the programmer remove one or both of the two existing methods, to reduce clutter.

A.2.3 Filtering Routines

At present two data filtering methods are included in Lrpiest. They are referred to as the *ad hoc* and LRPI filters. Only one filter may be used for a given run. Choice of the filtering method is made, as for the forgetting method, through the EstON parameter. Any value less than 10 for EstON specifies LRPI and a value greater than or equal to 10 chooses the *ad hoc* filter.

ad hoc filtering

The *ad hoc* filter is applied in the subroutine pre_filter. The filter applied is of the form:

$$y_f(t) = \frac{Tn0 + Tn1q^{-1} + Tn2q^{-2}}{Td0 + Td1q^{-1} + Td2q^{-2}} y(t)$$

Old values of the filtered signals: y_f, u_f etc. are retained between samples, so the actual implementation is:

$$y_f(t) = \frac{(Tn0 + Tn1q^{-1} + Tn2q^{-2})y(t) - (Td1 + Td2q^{-1})y_f(t-1)}{Td0}$$

For identification of a model suitable for GPC, it is necessary that the Tn terms contain a differencing factor, $1 - q^{-1}$. The Td terms are usually low-pass in form, for example Td0=1, Td1=-1.6, Td2=0.64.

LRPI filtering

The file calclrpi.c contains all of the routines to perform the LRPI filtering. LRPI requires the data filtering to be of the form:

$$y_f(t) = \frac{L(q^{-1})\Delta}{T(q^{-1})} y(t)$$

where Δ and $T(q^{-1})$ are imposed by the GPC disturbance model and $L(q^{-1})$ is the LRPI filter. The controller T filter is accessible through the data table mnemonics t1 and t2. The filter is applied in much the same way as the *ad hoc* filter:

$$y_f(t) = (L(q^{-1})\Delta y(t)) - t1y_f(t-1) - t2y_f(t-2)$$

More specifically, the function apply_LRPI_filter calls the function calc_lrpi_filter, and then applies the resulting $L(q^{-1})$ filter.

Within calc_lrpi_filter, all calculations are performed in double precision. This was done because the spectral factorization method used makes use of deconvolution, which can result in a significant loss of precision. All intermediate values are

therefore stored as double precision variables. By convention, all calculations in C are performed in double precision, even if the result is single precision, so the use of double precision variables does not result in a major increase in the computational load.

The $E_{N_2}(q^{-1})$ polynomial is calculated within calc_lrpi_filter, and then the sum of the $E_j(q^{-1})E_j(q^{+1})$ polynomials is formed for $j = N_1$ to N_2. The $L(q^{-1})$ polynomial is calculated by the function "factor" which performs the spectral factorization using the method of Bohm *et al.*. The method used is discussed in considerable detail in Chapter 5, and the reader is directed thereto for a more detailed description.

Appendix B

MATLAB Files

B.1 Introduction

This chapter contains a listing of many files used to evaluate LRPI and a number of different forgetting methods. For the sake of brevity, not all of the script files used are included here. The files that are included are intended for the user to evaluate the methods for himself.

The script files, or ".m files" as they are colloquially known, may be divided into two groups: those associated with LRPI and those used for evaluation of the different forgetting methods.

B.2 LRPI .M Files

B.2.1 Batch Calculation

The following routine will calculate the optimal parameter estimates for a first order model to match a set of batch data. The parameter "steps" is a vector containing the predictions to be considered, e.g. steps=1 means just one-step-ahead; steps=[1;2] means 1 and 2 step ahead predictions, etc. Newton-Raphson is the method used,

and the plant is Rohrs' third order plant discussed throughout the thesis. The function igpc.m called at the end is a .m file written by C. Mohtadi and included in Mohtadi (1990).

```
%
%     Newton(theta,epsi,n,dt,steps)
%
%     theta - column vector of initial parameter estimates
%     epsi  - scalar tolerance on theta.
%     n     - number of time steps in sample run
%     dt    - control interval
%     steps - row vector containing the future predictions
%             of interest (e.g.: 10, or 1 to 10, etc.)
%
%     newnew  .m file to examine effect of using a multistep prediction
%     version of LS. The plant is Rohrs' third order example, the
%     model is first order. The attempt is to make the predictions
%     better over ''steps'' steps ahead
%
%                 D.Shook, March 28 1989
%
%
num=229*2;
den=conv([1 1],[1 30 229]);
[a,b,c,d]=tf2ss(num,den);[a,b]=c2d(a,b,dt);
[tfnum,tfden]=ss2tf(a,b,c,d,1);
Aold=[1 -.8922];Bold=[0 .2423];
y=dlsim(tfnum,tfden,u);
if inc==1,
      uu=[0;diff(u)];
      y=[0;diff(y)];
else
      uu=u;
end;
l=length(y);
for i=1:max(steps),
      uold(:,i)=[zeros(i,1);uu(1:l-i)];
      yold(:,i)=[zeros(i,1);y(1:l-i)];
end;
e=100;
while e > epsi,
    a=theta(1);
    b=theta(2);
    num=[0 b];
    den=[1 a];
%
    d2ydb2=zeros(uu);
    yhat=zeros(length(y),length(steps));dyda=yhat;dydb=yhat;
                                        d2yda2=yhat;d2ydab=yhat;
    for jj=1:length(steps),
       j=steps(jj);
       yhat(:,jj)=(-a)^j*yold(:,j);
       if(j>=2),
          d2yda2(:,jj)=d2yda2(:,jj)+j*(j-1)*(-a)^(j-2)*yold(:,j);
          for i=1:j-1,
             d2ydab(:,jj)=d2ydab(:,jj)-i*(-a)^(i-1)*uold(:,i+1);
             d2yda2(:,jj)=d2yda2(:,jj)+b*i*(i-1)*(-a)^(i-2)*uold(:,i+1);
          end;
```

```
        end;
        for i=0:j-1,
           yhat(:,jj)=yhat(:,jj)+(-a)^i*b*uold(:,i+1);
           dydb(:,jj)=dydb(:,jj)+(-a)^i*uold(:,i+1);
        end;
        dyda(:,jj)=-j*(-a)^(j-1)*yold(:,j);
        err(:,jj)= yhat(:,jj)-y;
    end;
    dyda=dyda+b*d2ydab;

    djda=sum(sum(err.*dyda));
    djdb=sum(sum(err.*dydb));
    djdt=[djda;djdb];

    d2jda2=sum(sum(err.*d2yda2))+sum(sum(dyda.^2));
    d2jdab=sum(sum(err.*d2ydab))+sum(sum(dyda.*dydb));
    d2jdb2=sum(sum(dydb.^2));

    hessian=[d2jda2 d2jdab; d2jdab d2jdb2];
    thetanew=theta-hessian\djdt
    J=sum(sum(err.^2))
    e=sum(abs(thetanew-theta))
    theta=thetanew;
end

t=1:20;t=t';
y1=dstep(tfnum,tfden,20);
y2=dstep([0 theta(2)],[1 theta(1)],20);
plot(t,y1,'-',t,y2,'+');
[rold,sold]=igpc(Aold,Bold,n1,10,1,0,1,1);
cpold=conv(rold,tfden)+conv(sold,tfnum);
oldrts=roots(cpold);
[rxct,sxct,cpexct]=igpc(tfden,tfnum,n1,10,1,0,1,1);
exctrts=roots(cpexct);

A=[1 theta(1)];B=[0 theta(2)];
[rnew,snew]=igpc(A,B,n1,10,1,0,1,1);
cpnew=conv(rnew,tfden)+conv(snew,tfnum);
newrts=roots(cpnew);
```

B.2.2 Recursive Calculations

For the recursive LRPI calculations, there were two main methods used: Recursive Gauss-Newton (RGN) and LRPI through adaptive fitlering (LRPI).

The following three functions simulate closed-loop LRPI and GPC control of Rohrs' plant. The function stoz is another file written by Dr. Mohtadi, and is a continuous-to-discrete function for transfer functions.

```
%
%       lrpigpc.m       closed loop LRPI and GPC.
%                       Horizons and orders in N
%
```

```
%
N=[1 1 1 1 10]; na=N(1); nb=N(2); N1=N(4); N2=N(5);
k=N(3); NU=1; lambda=0; p=1;
theta=[zeros(1,na) zeros(1,nb)]';
theta(na+1)=1;  th=[]; Lr=[];
P=10*eye(na+nb);
T=25*[1 -.8]; nt=length(T);
T2=[1 -.8];
D=[1 -1]; L=1;
y=zeros(50,1); u=zeros(50,1); ly=length(y); lu=length(u);
uf=u;yf=y; beta=1;
as=conv([1 1],[1 30 229]);
bs=458;
h=0.05;
[B,A]=stoz(bs,as,h,0);
nA=length(A); nB=length(B);
lw = length(w);
i=0; yr=[]; ur=[];
clc; clg;
while i < lw,
home,
i = i + 1,
ynew  = B(2:nB)*u(1:nB-1) - A(2:nA)*y(1:nA-1),
y     = [ynew; y(1:ly-1)];
if nt > 2,
  ufnew = (u(1) - u(2) - T(2:nt)*uf(1:nt-1))/T(1);
  uf    = [ufnew; uf(1:lu-1)];
  yfnew = (y(1) - y(2) - T(2:nt)*yf(1:nt-1))/T(1);
  yf    = [yfnew; yf(1:ly-1)];
else,
  ufnew = u(1)-u(2);
  uf    = [ufnew; uf(1:lu-1)];
  yfnew = y(1)-y(2);
  yf    = [yfnew; yf(1:ly-1)];
end;
[theta,P] = rlrpi(theta,P,N,L,yf,uf,beta);
theta,
ahat = [1 theta(1:na)'];
bhat = [zeros(1,k) theta(na+1:na+nb)'];
[r,s]= igpc(ahat,bhat,N1,N2,NU,lambda,T2,p);
nr=length(r); ns=length(s); nt2=length(T2);
if i >= nt2
     tw = T2*w(i:-1:i-nt2+1);
else,
     tw = T2(1:i)*w(i:-1:1);
end;
unew = (-r(2:nr)*u(1:nr-1) + tw - s*y(1:ns))/r(1);
if unew > 100, unew = 100; end;
if unew < -100, unew = -100; end;
lu = length(u); u = [unew; u(1:lu-1)];
L  = eqivl(ahat, D, T, N1, N2);
yr(i) = y(1); ur(i) = u(1);
th(:,i) = theta; Lr(i,:)=L;
end


function [theta,P]=rlpri(theta,P,N,L,y,u,beta)
%
%    rlrpi: filter y, u with L and then perform one RLS iteration.
%
na=N(1); nb=N(2); k=N(3); N1=N(4); N2=N(5);
```

```
nl=length(L);

for  i=1:na,
      phi(i)=-L*y(i+1:i+nl);
end;
for  i=1:nb,
      phi(i+na) = L*u(i+k-1:i+k-1+nl-1);
end;
phi   = phi(:);
e     = L*y(1:nl) - phi'*theta;
P     = P - P*phi*phi'*P/(1+phi'*P*phi); P=P/beta;
theta = theta + P*phi*e;
return


function [L]=eqivl(a,D,T,N1,N2)
%
%    eqivl: find L for a given model and controller tuning.
%
E=filter(T,conv(a,D),[1 zeros(1,N2)]);
m=conv(E(1:N1),E(N1:-1:1));
for i=N1+1:N2
      mi=conv(E(1:i),E(i:-1:1));
      m = [0 m 0] + mi;
end;
L=spctr(m);
L=L/sum(L);
return
```

B.3 Forgetting .M Files

Four different forgetting factors were investigated: those of Hägglund (1983), Ydstie *et al.* (1985), Kulhavý (1987) and Sripada and Fisher (1987).The following functions show how the RLS calculation is modified for each forgetting method.

Ydstie's Forgetting Factor

```
function [P,theta]=rlsyff(P,theta,phi,y,yhat,sigma,N0,emin)
%
%     rlsyff.m    recursive least squares (using Ydstie's
%      forgetting factor).
%
%     [P,theta]=rlsyff(P,theta,phi,y,yhat,sigma,N0,emin)
%
%     P           covariance matrix
%     theta       parameter estimate vector (vertical)
%     phi         regressor vector (vertical)
%     y           new measurement
%     yhat        a priori estimate of y
%     sigma       noise standard deviation
```

```
%     N0          asymptotic memory length
%     emin        prediction error deadband
%
e=y-yhat;
if abs(e) > emin,
      G=phi'*P*phi;
      lambda=yff(G,e,sigma,N0);
      P=(P - P*phi*phi'*P/(lambda+G))/lambda;
      theta=theta+P*phi*e;
end;
end;


function lambda=yff(G,e,sigma,N0)
%
%     yff.m       Ydstie's forgetting factor
%
%     lambda = yff(P,phi,e,sigma,rho)
%
%     lambda      exponential forgetting factor
%     G           phi'*P*phi
%     e           a priori prediction error
%     sigma       noise standard deviation
%     N0          asymptotic memory length
%
m=1-G-e^2/(sigma^2*N0);
lambda=(m+sqrt(m^2+4*G))/2;
end
```

ILS of Sripada and Fisher

```
function [P,theta]=rlsils(P,theta,phi,y,yhat,trp,emin)
%
%     rlsils.m    recursive least squares (using Sripada and
%         Fisher's forgetting factor).
%
%     [P,theta]=rlsils(P,theta,phi,y,yhat,trp,emin)
%
%     P           covariance matrix
%     theta       parameter estimate vector (vertical)
%     phi         regressor vector (vertical)
%     y           new measurement
%     yhat        a priori estimate of y
%     trp         desired trace
%     emin        prediction error deadband
%
e=y-yhat;
if abs(e) > emin,
      Pphi=P*phi;
      G=phi'*Pphi;
      r=G+1;
      lambda=1- 0.5*(r-sqrt(r^2 - 4*Pphi'*Pphi/trace(P)));
      P=(P - P*phi*phi'*P/(lambda+G))/lambda;
      theta=theta+P*phi*e;
end;
end;
```

Kulhavý's Directional Forgetting Factor

```
function [P,theta]=rlsdff(P,theta,phi,y,sigma,rho,delta)
%
%     rlsdff.m    recursive least squares (using
%         directional forgetting factor of Kulhavy, etc).
%
%     [P,theta]=rlsdff(P,theta,phi,y,sigma,rho,delta)
%
%     P           covariance matrix
%     theta       parameter estimate vector (vertical)
%     phi         regressor vector (vertical)
%     y           new measurement
%     sigma       noise standard deviation
%     rho         Kulhavy's ad hoc tuning parameter
%     delta       vector containing two elements:
%        delta(1)    minimum value of phi'*P*phi for update
%        delta(2)    minimum value of alpha for covariance update
%
e=y-phi'*theta;
G=phi'*P*phi;
en2=e^2/(sigma^2*(1+G));
lambda=1/(1+(1+rho)*(log(1+G)+G*(en2-1)/(1+G)));
alpha=lambda-(1-lambda)/G;
%
if G > delta(1),
       theta=theta+P*phi*e/(1+G);
       if abs(alpha) > delta(2),
            P=P - P*phi*phi'*P/(1/alpha+G);
       end;
end;


function [alpha,lambda]=dff(G,e,sigma,rho)
%
%     dff.m       Directional forgetting factor of Kulhavy, etc.
%
%     [alpha,lambda]=dff(P,phi,e,sigma,rho)
%
%     alpha       directional forgetting factor
%     lambda      exponential forgetting factor
%     G           phi'*P*phi
%     e           a priori prediction error
%     sigma       noise standard deviation
%     rho         Kulhavy's ad hoc tuning parameter
%
en2=e^2/(sigma^2*(1+G));
lambda=1/(1+(1+rho)*(log(1+G)+G*(en2-1)/(1+G)));
alpha=lambda-(1-lambda)/G;
```

Hägglund's Directional Forgetting Factor

```
function [theta,P]=hag(theta,P,phi,y,a,v)
%
%     hag.m       Directional forgetting factor of Hagglund, etc.
%
```

```
%     [theta,P,alpha]=hag(theta,P,phi,y,a,v)
%
%     alpha       directional forgetting factor
%     theta       vertical parameter vector
%     P           covariance matrix
%     phi         vertical regressor vector
%     y           new observation
%     a           tuning parameter: eventually P=a*I
%     v           tuning parameter: desired variance of residual
%
G=phi'*P*phi;
S=P*phi;
deltad=(S'*P*S/(S'*S)-a)/(S'*S);
alphad=inv(v)+deltad/(deltad*G-1);
%
if alphad<=0
       alpha=0;
elseif alphad<=inv(G),
       alpha=alphad;
elseif alphad<=(inv(v)+inv(G)),
       alpha=inv(G);
elseif alphad>(inv(v)+inv(G)),
      alpha=0;
else
     disp([' weird error in hag.m'])
end
%
alpha=alpha;
K=S/(v+G*(1-alpha*v));
theta=theta+K*(y-phi'*theta);
%pden1=inv(inv(v)-alpha);
%pden2=inv(pden1+G);
%P=P-S*S'*pden2;
P=P- S*S'/((inv(inv(v)-alpha))+G);
```